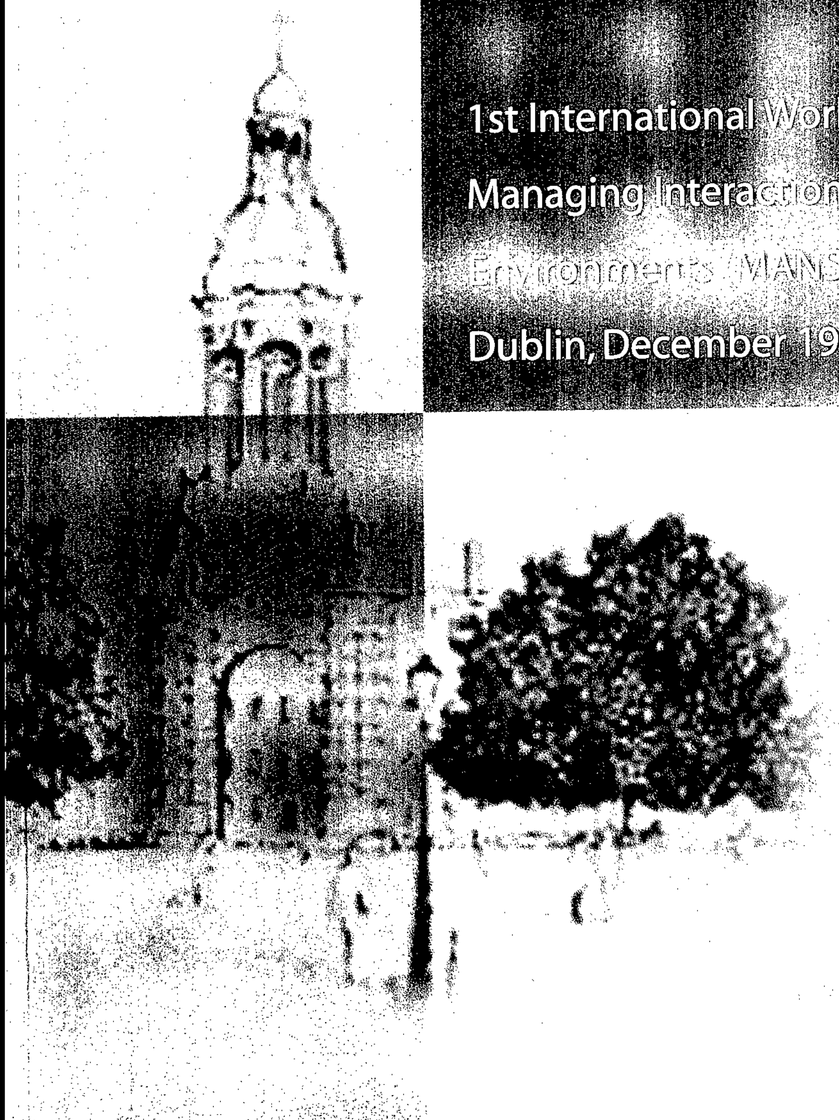
Paddy Nixon, Gerard Lacey and Simon Dobson (Eds)

Managing Interactions in Smart Environments

1st International Workshop on Managing Interactions in Smart Environments (MANSE'99), Dublin, December 1999



REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE December 1999	3. REPORT TYPE AND DATES COVERED Conference Proceedings

4. TITLE AND SUBTITLE 1st International Workshop on Managing Interactions in Smart Environments (MANSE 99)	5. FUNDING NUMBERS F61775-00-WF
6. AUTHOR(S) Conference Committee	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Trinity College, Dublin xxx Dublin Ireland	8. PERFORMING ORGANIZATION REPORT NUMBER N/A
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) EOARD PSC 802 BOX 14 FPO 09499-0200	10. SPONSORING/MONITORING AGENCY REPORT NUMBER CSP 00-5004

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	12b. DISTRIBUTION CODE A

13. ABSTRACT (Maximum 200 words)

The Final Proceedings for 1st International Workshop on Managing Interactions in Smart Environments (MANSE 99), 13 December 1999 - 14 December 1999

This is an interdisciplinary conference. The conference will address the convergence of research in Distributed Systems, Robotics and Human Centered Computing within the domain of smart buildings and present a unique opportunity to investigate work that crosses the boundaries of these disciplines.

14. SUBJECT TERMS EOARD, Sensor Technology, Human Factors, Image Processing, Distributed sensors, knowledge bases	15. NUMBER OF PAGES 250
	16. PRICE CODE N/A

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19, SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18
298-102

Managing Interactions in Smart Environments

Springer
London
Berlin
Heidelberg
New York
Barcelona
Hong Kong
Milan
Paris
Santa Clara
Singapore
Tokyo

Paddy Nixon, Gerard Lacey and Simon Dobson (Eds)

Managing Interactions in Smart Environments

1st International Workshop on Managing Interactions in Smart Environments (MANSE '99), Dublin, December 1999

Sponsors

Organised by
Department of Computer Science,
University of Dublin, Trinity College

Springer

Paddy Nixon, BSc, MA, PhD, CEng, MBCS
Gerard Lacey, BA, BAI, MA, PhD
Simon Dobson, BSc, DPhil, CEng, MBCS
Department of Computer Science, Trinity College Dublin, Dublin 2, Ireland

ISBN 1-85233-228-X Springer-Verlag London Berlin Heidelberg

British Library Cataloguing in Publication Data
Managing interactions in smart environments : 1st
International Workshop on Managing Interactions in Smart
Environments (MANSE '99), Dublin, December 1999
1.Human-computer interaction - Congresses
I.Nixon, Paddy II.Lacey, Gerard III.Dobson, Simon
IV.International Workshop on Managing Interactions in Smart
Environments (MANSE '99) (1st : 1999 : Dublin)
004'.019
ISBN 185233228X

Library of Congress Cataloging-in-Publication Data
A catalog record for this book is available from the Library of Congress

Typesetting: Camera ready by contributors
Printed and bound by the Athenæum Press Ltd., Gateshead, Tyne & Wear
34/3830-543210 Printed on acid-free paper SPIN 10746810

Foreword

The embedding of computational intelligence into the objects of our daily lives has been made feasible by recent developments in computer hardware design. The citizen of the future has the prospect of having their clothes, their personal digital assistant and their home, communicate via a wireless network. We can see from the next generation of mobile phones that the Internet and multimedia technology will be truly mobile and ubiquitous. This presents great opportunities for new applications and facilitates new ways of interacting with computers. It also challenges researchers to find new ways of developing and managing these dynamic and diverse systems.

The ubiquity of network access and power of embedded processors will allow the customisation of systems to meet your personal preferences no matter where you are: The radio in your Bangkok hotel room will automatically tune to your favourite web radio station, your personal agent recommends a list of good local restaurants with tables available, the room recognises that you are sleeping and so filters your mobile phone calls except those from your kids. The possibilities for applying this technology are limited only by our imaginations.

In this first MANSE workshop we are concerned with managing the storm of interactions within smart environments and during the workshop we intend to demonstrate some potential applications in our own building, the O'Reilly Institute. Interdisciplinary work is very important and it is particularly pleasing that the workshop is being organised by two of our leading research groups, the Computer Vision and Robotics Group and the Distributed Systems Group. Such a workshop is a fitting way to mark the thirtieth anniversary of the Department of Computer Science at Trinity College, Dublin.

The papers which have been selected by the international programme committee aim to define a new interdisciplinary domain within Computer Science. This collection of papers represents some of the highest calibre industrial and academic research from both Europe and America. I hope that this will be the first of a series of MANSE workshops and that it will be a particularly enjoyable and creative one for the attendees.

John G. Byrne
Professor of Computer Science
Department of Computer Science
School of Engineering
Trinity College Dublin
Ireland

Acknowledgement

The editors would like to thank the following organisations and people for their support of MANSE '99. Firstly we would like to thank our sponsors without whom this conference would have been impossible to run:

- European Union TMR Network: Semi-Automonous Monitoring and Robotics Technology (SMART II).
- European Union TMR Network: Mobile Robotics Technology for Health Care Services Research Network (MobiNet).
- European Office of Aerospace Research and Development, Air Force Office of Scientific Research, United States Air Force Research Laboratory.

We would also like to thank Richard Greenane for maintaining the web page and all the authors and reviewers who worked so hard to make this workshop a success.

October 1999 Paddy Nixon, Gerard Lacey, Simon Dobson

International Programme Committee

List of authors

Gregory Abowd
Georgia Tech, US

Sumit Basu
MIT, US

Michael Beigl
University of Karlsruhe, DE

Aaron Bobick
Georgia Tech, US

Vinny Cahill
Trinity College Dublin, IE

Michael Coen
MIT, US

Rem Collier
University College Dublin, IE

James Crowley
Inria Rhône-Alpes, FR

Per Dahlberg
The Viktoria Institute, SE

Anind Dey
Georgia Tech, US

Simon Dobson
Trinity College Dublin, IE

Brian Duffy
University College Dublin, IE

Guillaume Durand
Inria Rhône-Alpes, FR

Lars Erik Holmquist
The Viktoria Institute, SE

Peter Finin
MIT, US

Grant Foster
University of Reading, UK

Michael Fox
MIT, US

Hans-Werner Gellerson
University of Karlsruhe, DE

Marion Groh
MIT, US

Mads Haahr
Trinity College Dublin, IE

O Habert
University of Metz, FR

Daniela Hall
Inria Rhône-Alpes, FR

W Hardin
University of Reading, UK

Eric Jul
DIKU, DK

Markus Lauff
University of Karlsruhe, DE

Stasha Lauria
University of Reading, UK

Christophe Le Gal
Inria Rhône-Alpes, FR

Peter Ljungstrand
The Viktoria Institute, SE

Jérôme Martin
Inria Rhône-Alpes, FR

Paddy Nixon
Trinity College Dublin, IE

Ruadhan O'Donoghue
University College Dublin, IE

Michael O'Grady
University College Dublin, IE

Gregory O'Hare
University College Dublin, IE

Ronan O'Rafferty
University College Dublin, IE

Alex Pentland
MIT, US

Stephen Peters
MIT, US

Brenton Philips
MIT, US

Claudio Pinhanez
IBM Research, US

Johan Redström
The Viktoria Institute, SE

Colm Rooney
University College Dublin, IE

Daniel Salber
Georgia Tech, US

Steve Shafer
Microsoft Research, US

Flavia Sparacin
MIT, US

Kavita Thomas
MIT, US

Tim Walsh
Trinity College Dublin, IE

Nimrod Warshawshy
MIT, US

Luke Weisman
MIT, US

Kevin Wilson
MIT, US

Christopher Wren
MIT, US

Bryant Yeh
MIT, US

Contents

Smart Environments: some challenges for the computing community

Paddy Nixon, Simon Dobson, and Gerard Lacey

Department of Computer Science, Trinity College, Dublin, Ireland

Building computing systems that exhibit intelligence, adaptability and seamless interaction is a challenging task in and of itself. It is a task that synthesises the expertise and concerns of a range of computing disciplines. The goal of such research, in or opinion, should be to move these intelligent, adaptable, heavily interacting systems into everyday life in such a way that they become as commonplace as other artifacts in our environment - until in effect they become essentially invisible to their users. Embedding such systems into the very environments we inhabit, rather than crafting them as tailored environments in their own right, posses a significant research challenge that can only be addressed in an interdisciplinary manner.

In this short paper we call for a holistic approach to the development of smart environments, as both an introduction and unifying theme to the detailed research contributions in this volume.

Motivation

Research into smart buildings and spaces has been steadily increasing over the last few years[1]. Active buildings promise to provide greater access, usability and usefullness to both able-bodied and disabled people at work or at home. The interactions involved in such systems include many heterogeneous elements: between systems and networks, between systems and systems, and between systems and people.

Smart environments can usefully perform a range of tasks: planning tasks, such as suggesting routes through the buildings to visitors; perception tasks, such as gesture and face recognition; action tasks such as calling lifts or robots; and software tasks such as managing the massive event reporting these real-world systems will generate. Facilitating these interactions requires the development of a variety of software infrastructures from support for actuators and sensors, through novel control and interaction interfaces, to distributed and mobile systems infrastructures. These supports cannot be developed in isolation from the problem domain, or from each other: indeed, it is imperative that we address this convergence of research in distributed systems, robotics and human centred computing within the domain of smart environments in a holistic way if we are to have any chance of providing focused and integrated solutions. This presents a significant research and development challenge to the computing community as whole[3].

Themes

Establishing the need for a holistic view of smart environment research gives us a framework for reviewing the area and for the structure of this volume. We divide the subject into five key technical areas:

- Interaction and control
- Learning and interpretation
- Robotics
- Programming
- Exemplars of intelligent environments

Taking each area in detail:

Firstly, we must be able to interact with our environments in intuitive and appropriate ways. Integral to this goal is the need to take computer interaction beyond keyboard and mouse and to facilitate interaction through normal tasks such as speech, hearing, vision, and gesture - and then integrate the actions of a number of participants with different contexts and modalities.

Providing an interaction mechanism is the foundation upon which intelligence can be layered, but static interfaces can have only limited adaptability to individual users. Environments must be able to learn interaction patterns, and in doing so improve the quality of the interaction between system and participant.

Robotics, and its associated research areas, both complement and are enabled by the smart environment. Nowhere is this clearer than in the design of independent living environments where robotic aids take advantage of the intelligence and reactive characteristics of an environment. The holistic approach promises to improve the quality of life and reduce costs of care immeasurably.

We have so far addressed humans' communication with the environment and the environments' responses. It is clear, however, that such fine-grained and adaptive interactions will be difficult if not impossible within traditional low-level programming languages and user interface methodologies. Making the resources of the active building available to applications (and vice versa) requires new programming models - ones which are not tied so closely to our keyboard/screen/mouse or client/server views of the programming world, but which take account of notions such as movement and trajectory within a physical or computation space. Equally, managing the software of the environments themselves differs significantly from more static situations and requires well-founded design models and automated support.

Finally, it is our contention that this synergy of subjects will not mature without benchmarks against which different approaches can be compared. As in any other engineering discipline, such benchmarks have to be designed carefully to ensure that exactly the features of interest are exercised and measured. An important aspect of this volume is the start of the debate, via a number of exemplars, on what constitutes an appropriate benchmark for intelligent environments.

To be or not to be?

Our own contribution to this debate is taking place within the Intelligent Interfaces to Buildings (IIB) project in TCD. The purpose of IIB is to provide a test-bed for work in a number of different areas represented within the Department, including:

Vision and augmented reality. Providing novels ways to integrate gesture and vision, with augmented reality views of both the physical and information space.
Intelligent agents. Embedding intelligence into the device via an intelligent agent view of the system.
Robotics in health care. Improving the function and cost effectiveness of assistive robotics by leveraging the features of intelligent environment.
Programming models. Building new programming languages for multi- modal systems, and directly supporting in the language and middleware key infrastructures requirements such as events, mobility, policies, and management.

Unifying this work is a desire to provide core infrastructure that allows different subsystems to interoperate in a well-understood manner, adding computational intelligence and suppprt to the participants' interactions both with the environment and with each other. We believe that this holistic, non-intrusive, pervasive approach will be key to the long-term maturation of smart environments.

Conclusions

The fundamental aim of research in intelligent environments is to provide a totally ubiquitous computing environment were people can work and play anywhere, at anytime, and in the manner that best suits them[4]. Mark Weiser, the originator of the phrase of Ubiquitous Computing, says it best when he describes the future and potential of ubiquitous computing:

> The technology of literacy when first invented, and for thousands of years afterwards, was expensive, tightly controlled, and precious. Today it effortlessly, unobtrusively, surrounds us. Look around now: how many objects and surfaces do you see with words on them? Computers in the workplace can be as effortless, and ubiquitous, as that. Long-term the PC and workstation will wither because computing access will be everywhere: in the walls, on wrists, and in "scrap computers" (like scrap paper) lying about to be grabbed as needed.[2]

This is a noble and ambitious goal, and one whose realisation requires more than focused research within a particular technical area. It requires us to embrace the physical environment and bring it into our computer systems. By doing so we shall hopefully develop both interesting and useful technology and some new perspectives on the relationships between computing and living.

References

1. MichaelCoen. *Design principles for intelligent environments*, in Proceedings of the AAAI Symposium on Intelligent Environments, pp. 36-43. AAAI Press. March 1998.
2. Mark Weiser. *The Computer for the Twenty-First Century.* Scientific American, pp. 94-104. September 1991.
3. Special issue of Communications of the ACM. *Computer Augmented Environments: Back to the Real World.* July 1993.
4. Paddy Nixon and Vinny Cahill (guest editors). Special issue of IEEE Internet Computing **2**(1). *Mobile Computing.* January/February 1998.

Ten Dimensions of Ubiquitous Computing

Steve Shafer

Vision Technology Group, Microsoft Research
Microsoft Corp., One Redmond Way, Redmond WA 98052 USA
http:// www. research.microsoft.com/easyliving

Abstract. Much has been written about smart mobile computing devices and how they will make "ubiquitous computing" a reality, but simply having smart devices in the world is not enough. An infrastructure is needed to bind these devices together in a meaningful way. This infrastructure is known as "intelligent environments". But, what are the elements of this "intelligence" behind ubiquitous computing? In this position paper, we propose ten dimensions along which this intelligence might be described. They can be summarized as: meaningfulness, world modeling, user modeling, distribution, accessibility, extensibility, heterogeneity, automation, usability, and ubiquity. The intelligent environment systems of today are demonstrations of subsets of this vision; the task before us is to generalize from these examples and develop the standards that will allow ubiquitous computing to be fully developed.

Elements of Ubiquitous Computing

The computing industry is undergoing a revolution away from the "personal computer" and towards a network of wired and wireless devices of varying characteristics, each with its own role to play. Figure 1 shows representatives of the key elements in this emerging paradigm of computing.

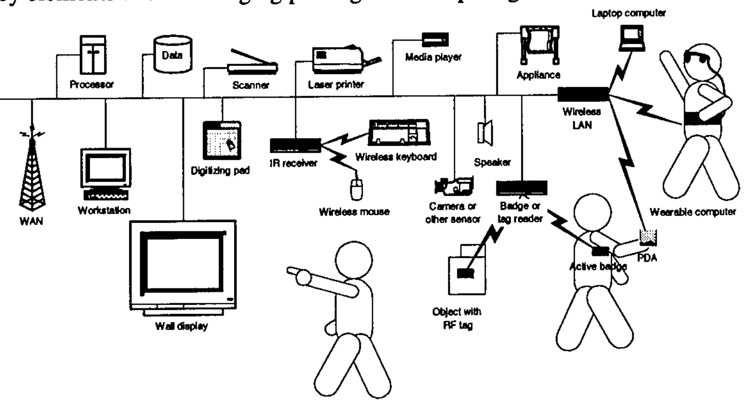


Fig. 1. Representative Elements of Ubiquitous Computing

These elements include:

- A central LAN connecting components of the infrastructure, with a WAN gateway.
- A wireless LAN connecting the LAN to other computers such as laptops, wearable computers, and handheld computers (palm computers, tablet computers, or PDAs).
- Workstations that are wired to the LAN and include many components, both interactive (display, keyboard, mouse, speaker, microphone) and non-interactive (CPU, memory, media drive).
- Peripherals for user interaction that are wired to the LAN, such as wall displays (or other displays or display projectors), digitizing pads, wired keyboards, speakers, microphones or microphone arrays, and mice; and also wireless interaction peripherals such as a wireless mouse or keyboard that can move about the room.
- Scanners, printers, media players, etc., which are peripheral devices that people use, though they are generally used in a non-interactive way (i.e. interaction cycle times may be many seconds long).
- Processors and data storage devices attached to the LAN. These may be resources for computing, but users do not directly physically interact with them.
- Sensors of a more specialized type, such as cameras, active badge sensing systems, or RF tag readers. These are more "researchy", being outside the standard GUI/WIMP and audio interface paradigms, and include both interaction devices (such as an active badge with buttons on it), and non-interaction devices (such as a thermometer in a room). Some devices may be used in both interactive and non-interactive ways, such as a camera.
- Computer-controlled appliances such as an oven, water faucet, door lock, or light socket.
- People in the environment, whether wearing a computer, using a computer or networked peripherals, using wireless devices, or simply going about their business without explicitly wearing, carrying, or touching any computing devices at all.

The emerging paradigm for computing includes all these elements. This comprehensive concept may be called "ubiquitous computing", a term articulated by Marc Weiser [weiser93]. We take the view here that the set of devices that are static and connected on the LAN form the "intelligent environment", constitutes an infrastructure framework for whatever mobile devices may be added through wireless communication. Together, the environment and the mobile elements form the ubiquitous computing system. It may be tempting sometimes to imagine that a single intelligent environment can exist without all the mobile computing elements of ubiquitous computing, but in this paper we express the belief that intelligence implies an ability to work in cooperation with mobile devices. Given that those same devices must also operate intelligently in other environments, it means that all intelligent environments must share some common interfaces, and thus we arrive at ubiquitous computing.

In this paper, we take the view that all of the elements shown in Figure 1 are part of the emerging nature of computing systems, and we are concerned with architectural infrastructure, and interaction paradigms, that give the person using these facilities the best possible experience. Some people believe that the best experience will be "invisible computing", in which the user may be unaware that there is a computer at all, but that is not the view we take in this paper. Rather, we seek an "appropriate computing" experience, in which both explicit and implicit interaction with the computing elements can be combined, to take full advantage of the available resources. Also, we are concerned here with concepts that are general in the sense that they apply to many different subsets of Figure 1; our view is that each person will have a different suite of mobile devices, and each space will have a different subset of static devices. In this paper, we will attempt to identify some of the characteristics that underlie all systems for ubiquitous computing, with emphasis on "intelligent environments", that is, rich LAN structures, as opposed to emphasizing wireless networking or worn computers.

Ten Dimensions of Ubiquitous Computing

Little has been said about what requirements must be met by an intelligent environment to realize the vision of ubiquitous computing. Here, we present ten "dimensions" or questions about intelligent environments that help to clarify the characteristics of such systems.

How Can Devices Interact Meaningfully?

The most obvious thing about ubiquitous computing or intelligent environments is that they have lots of devices talking to each other. Accordingly, one of the first questions in building such systems is how to get these things to talk to each other at all. This raises questions of network protocols, distributed object programming systems, etc, and much research proceeds along these lines, as though that were sufficient for ubiquitous computing. But, assuming the connections are made somehow, the deeper question is, how can we make these interactions *meaningful*? What would it take to call device interactions meaningful?

One essential element is that some things have to be automatic. If nothing is automatic, the mere fact that each device is capable of talking to one or more others is immaterial, there are simply too many of them and the user will be unable to cope. Instead, we need to have some notion of devices acting in reasonable combinations, where the system itself somehow helps the user figure out what combinations are reasonable. For example, when I have a laptop computer configured to print on my home printer, and I carry that computer to my office, it's at the limit of my ability to remember to manually reconfigure the computer for the office printer. If my home and office are "intelligent environments" with many computing devices, I couldn't possibly make all the necessary changes every time I lug my laptop around. It's enough that I do the carrying – I will need help from the system to connect the right things to the right things for me. But, how much is automatic, and how much should be automatic?

Another characteristic of meaningful interaction is that devices have to interact with each other and with people. Interacting means more than just being capable of executing code, it means understanding the role of each device, and the activity of the world and the people in the world, and causing the system to act accordingly. If I am working by myself, it's fine with me for the system to announce incoming messages with a beep and a flash, but if I am talking to other people, then I don't want to be interrupted by any message that is not urgent. I want the system to figure this out and do something reasonable.

How much do devices have to know in advance about each other? If the behavior of each device has to be hand-coded based on the known locations and characteristics of other devices, then I'm going to have a very hard time adding one new device, or moving one device to a new position, or wiring up another room in my house. So, we need automatic discovery of devices, and we need automatic adaptation of behavior based on the current configuration and state of the devices in the system.

Given such a system of explicit reasoning about devices and people and the world, how is an application program supposed to make things happen? The current paradigms for computing call for applications to create graphics and sounds and send them directly to specific devices. But, in the world of ubiquitous computing, the very definition of what device is correct may change from moment to moment, and as a person moves from place to place, the set of available devices and their characteristics may vary. So, the paradigm for applications to communicate with users may have to be substantially different from the current paradigm. The need will be both for multimodal interfaces using new devices, and also for interfaces that can be expressed on a variety of different devices, perhaps from moment to moment.

In these few paragraphs we have summarized many of the key issues; in the next sections, we will explore these in more detail.

What Is Known About The World?

Intelligence requires knowledge, and intelligent environments have to know things about the world. How much do they need to know, and how will they acquire this information?

One limit to the world knowledge comes from the suite of available sensors. Many sensors directly measure things of interest. Sensing is relatively reliable and specific sensors are inexpensive and can measure many different things. So, there is a school of thought that says, why not directly sense everything of interest? The problem is that such sensors require networks of wiring for their deployment, and because they have no generality, there is a combinatorial explosion in the number of sensors needed to know lots of things in lots of places. In addition, sensor data can sometimes be easily fooled, for example a pressure sensor in a chair seat may register for a person sitting there, but also for a heavy object placed in the chair.

Robotics distinguishes between sensing, which is direct measurement of properties in the world, and perception, which involves reasoning about sensed data to derive information. In perception, such as computer vision, a few very general-purpose sensors are deployed to gather data over a broad area, and then elaborate algorithms are used to try to derive meaningful information from this mass of data. Computer vision is notoriously bad at determining very general information, so in recent years attention has turned more towards special purpose algorithms for extracting specific facts from the sensory data. Much good progress has been made, and indeed many of the most "intelligent" of the intelligent environments demonstrated to date have been based on computer vision as the primary means of sensory input. However, progress in this area is quite slow. Also, this paradigm raises an interesting limit to the world knowledge available to the system – it is limited by the repertoire of perceptual algorithms available. This limit can be quite sharp, because most algorithms to date work only in limited conditions, so that even a good perception algorithm may only be usable on a fraction of the sensors in the environment.

The paradigm of perception also raises a question – How do you notice things? With a dedicated sensor, as long as it is on, it will report events or measurements. But, using perception algorithms, you are guaranteed not to notice something unless you are currently running the specific algorithm to detect that thing. Therefore, to conserve resources, you have to choose what set of things you will attend to, and ignore the rest. Perception requires a mechanism for the invocation of perceptual algorithms as appropriate.

Whatever you know about the world, meaning both state description and events, some representation is required to communicate this knowledge from sensors to applications and other system elements. Thus, an ontology of knowledge is required. At present, we are just beginning to see what belongs in this ontology.

Finally, the system needs to have knowledge about itself, about the capabilities of its various devices, their states, what is known versus what is inferred, what limits of precision or reliability pertain to the data, etc. And, for controllable devices, the system needs to know at least what controls are available, and perhaps some way to reason about the consequences of invoking those controls.

What Is Known About People?

Several different aspect of modeling people are important for intelligent environments. First, what body properties can be sensed? Stand-off sensors such as cameras and microphones provide some information, but usually a good deal of perception is required to infer useful data from such sensors. Depending on the models available (e.g. the kinematics of the body shape model), varying amounts of data may be represented. Some properties can be measured best by worn sensors, raising additional issues of communication and integration of this information. Authentication (i.e. determining identity) is a special example of knowledge about people, which can be sensed or inferred.

Another issue concerns mobility – What data moves with the person, and what computational capability moves? Two quite different paradigms have emerged for representing information about people – one postulates a central repository in which information is looked up based on the person's identity; and the other postulates that people carry with them a portable data store containing all personal information. The former approach makes all data available whenever needed, the latter provides some direct control over the availability of personal information. Neither is a clear winner to date. Also, most intelligent environments simply assume that the person has no mobile computational ability; in reality, many people carry at least a PDA today, so some representation is needed for this to allow the environment to take advantage of such devices when available.

Privacy becomes a sharper issue as well, when the system is monitoring your activities moment by moment. The system may need some information for optimal functioning, such as a model of what you look like and how you speak; but perhaps you wish to reveal this without at the same time revealing everything else about yourself. How can you specify the boundaries for various types of information about you? And, how do you even know what the system knows about you?

Where Does Information Reside?

Ubiquitous computing systems involve many computers. This raises the question of where data resides, and how it can be located. There is a need to locate not only data *per se*, but also computations such as I/O brokers that can route messages as appropriate. If such services are to be utilized by a broad range of applications, some degree of standardization will no doubt be required.

What information migrates from place to place on the network? And, what migrates from place to place in the physical world? These two are complementary questions. For example, based on resource availability, a distributed system may choose to migrate some computation or data store to a different processor; as long as it can be located by all who need it, who cares? On the other hand, if a person carries a mobile data store from place to place, it may move in the physical world but retain the same network characteristics and therefore be available electronically in an unchanging manner. Some who propose that people carry personal data stores presume that they will be available only in a limited way based on the physical location of the person, but if the data store presents itself to the local network, such limits may be hard to implement.

How are mobile and static elements to be integrated? Many intelligent environment demonstrations of today do not attempt such integration. And, many mobile computing scenarios do not presume any integration with a locally available LAN. Both are extreme positions. An intelligent environment ought to be intelligent enough to recognize not only its own devices, but also those that are carried in by people; and a mobile system ought not be forced to assume that it is all alone in the universe as the sole organ of I/O and computation. People use the resources in their space; so should mobile computers.

The issues of information residency are not only about mobility, some also have to do with permissions and access to resources. For example, suppose you buy a camera and a computer and some computer vision software. Later, you buy some new computer vision software from a different party. Is that new software allowed to be resident on the computer that came with the camera? If so, then the system would need some way to prioritize between the previous software and the new software which is now also competing for resources. At what level would standardization be needed to allow such sharing of hardware?

Who Can Access What?

A more general form of the access question is: Who can access what? "Who" could refer to people, to system software, or to applications; "what" could refer to programs, data stores, and hardware resources. The two faces of the question are how to grant access and how to deny access.

Granting access first requires some mechanism for discovery of the resources that are available. Some systems postulate a central directory service; others postulate that new arrivals announce themselves. In either case, an ontology is needed to standardize the descriptions of resources. In addition to the directory or announcement service, a naming mechanism is also needed to allow persistent names for resources to be discovered, stored by others, and passed from one program to another. When resources are allowed to migrate within the system, such naming mechanisms can be very challenging to implement.

Another critical issue is what information is stored in the resource directory (or contained in announcements). In most current designs, the information stored is simply a name for the resource and a type description. This is rather analogous to having a yellow pages telephone book with only phone numbers but no addresses – you can discover a list of resource providers, but you have no idea which ones would make more sense for you given your own situation. Other systems have begun to appear in which there is also geometric information in the form of a world model, allowing lookup of information based on location and facing.

Restricting access is just as important as granting it. One role of access restriction is preventing every program from having to attend to every event on the network, which is necessary when the network becomes complex. For example, if a geometric model is available, then programs could register their interest in events within some geometric area (such as the area in front of a display screen, for example).

Another role of access restriction is providing different privileges to different customers. This has manifestations both in the physical and electronic domains. In the physical world, access restriction may mean that there are some boundaries defining domains of administrative responsibility for devices and computers, for example "ownership" of a room or a building. Access privileges might be assigned within each domain, and granted based on the domain identities of the inquiring agent. In the electronic world, programs or people might be assigned to organizational domains; as before, information could be associated with these domains, and access privileges might then be assigned to the domains and granted based on the domain identities of the inquirer. In any case, there may be many such domains to be defined and assigned privileges; how is all that to be made manageable?

How Does The System Grow?

Any intelligent environment or ubiquitous computing system must allow for growth and improvement of its constituent parts, or it will die. This growth can occur along several dimensions:

- Additional devices: When a new device is added to the system, how is it announced, calibrated, and administered? Hopefully, it can immediately be used as a full partner in the system, without requiring reprogramming of the applications already resident; this requires that the system be able to reason explicitly about the nature, capabilities, location, and interface of the new device.
- Regions of coverage: Suppose a new set of devices are added that provide coverage of new areas, for example the corners of a room, or a new hallway, or a new room adjoining a hallway. How will the system represent this new area, so that services can be provided using the new devices?
- I/O modalities: New devices and paradigms are being developed constantly, for example 3D holographic displays, 3D "mice", and body sensors. Such new modalities can be used in two ways: by taking advantage of their new and unique characteristics, or by emulating well-understood standard modalities such as keyboard, 2D display, or mouse. How can the system allow new modalities to be incorporated and utilized in both of these ways?
- World model: The ontology that comes with the system software will no doubt be limited to the best-understood and most standardized devices etc. Every time you acquire a new device, application, or perception program, more will need to be added to that ontology. So, the ontology itself must provide some mechanism for its own extension. How can this be done?
- Application software: Applications will not all be written by the device manufacturers, nor by the system software provider, nor by the builder or owner of the room. For ubiquitous computing to be a commercial reality, it must be possible for third parties to write applications that will work across a variety of hardware and software configurations. What is needed to make this possible?

Ultimately, all of the "backbone" components that link together the ubiquitous computing elements must be scalable, to a high degree. Scalability is not something that is added *ex post facto* to a system, rather it must be a design consideration from the earliest days, and must be applied to each element of the system. Only then can growth be smoothly accommodated.

How Does Software Adapt To Hardware?

Consider a program that is about to begin execution in a ubiquitous computing system. Presumably, we know the person who has invoked the program. The program needs to present its interfaces on appropriate devices. This raises several questions. First, how does the program figure out what devices are available to the person? This may require some lookup mechanism, and possibly also some assessment of the person's preferences and situation. The application programmer cannot know in advance exactly what device will be used, and perhaps not even the modality (e.g. graphics v. speech). How can the program present an interface that can be expressed on a variety of different devices?

Even when devices correspond to the same modality, their characteristics may differ, for example the keyboard on a computer v. the "software keyboard" on a PDA. Some PDAs can express output only, for example some handheld calendar organizers that have no input capability. Software that is updating a person's calendar may have to contend with this characteristic. In fact, that same software might be called on to simultaneously update the calendar display on a monitor and the calendar data on the PDA, so that it may not even be known to the programmer how many devices should be updated by the software.

This discussion should properly not be limited to actual devices, but should extend to "virtual devices" as well. For example, it is clear that a monitor presents a display surface. But, a projector aimed at a space on the wall also presents the potential for a display surface, and the semantics are almost identical to those of the monitor. Input devices are even more mind-boggling, for example a "pointer" might correspond to a mouse, a touch screen, a camera observing a person point at a display monitor, or even a camera observing a person pointing at a blank wall spot with a projector aimed at that same area!

Probably, all of the reasoning implied here is too much to expect from the application programmer. Somehow, we need to define a new paradigm of interface between programs and computing devices that allows the system to account for mobility, variation of modalities (or even mixture of modalities), dynamic device selection, etc.

How Can We Specify Behavior?

Suppose we want to sit down and read a book in an intelligent environment. We need to have some light. How might we tell the environment to turn on the lights?

- Flip a switch that is wired, by hardware or software, to the lights. This is the most direct way, but we can also do it in unintelligent environments.
- Call up a dialog box on a convenient display and poke the button to turn on the lights. Requires some system intervention, but not especially intelligent.
- Call up a map showing the room and lights, and poke the one to turn on. Clunky, but fun.
- Say, "Turn on the overhead light for me." Here's a different modality, interfacing to the same device.
- Make a gesture (point at the light, perhaps?). This requires a camera, a good deal of perception, and a suitable gesture vocabulary. Each person probably makes the gesture somewhat differently, so there is a considerable effort needed to implement this.

- Say, "Give me light!" and expect the system to turn on the correct light. This would require some degree of reasoning as well as speech recognition and interpretation.
- Sit down and expect the system to turn on the light automatically. Now, *this* is intelligent!

It is the view in this paper that an intelligent environment ought not only to support the last method, but all of the preceding ones as well. People like to control things directly, and even when control is automatic, sometimes people like to override that automatic behavior with explicit directions to the contrary (for example, if the system hangs or has an annoying behavior). Automatic behavior can complement direct control, without necessarily replacing it. Note that this view stands in contrast to that of "invisible computing", a school of thought stating that computing interfaces ought to always be implicit and not explicit. In the above list, however, all but the first and last methods involve explicit communication with the computing system. Therefore, we contend that effective interfaces do not always require that computing be invisible.

This last method of turning on the lights, which *is* truly that of "invisible computing", assumes that the environment has some stored behavior specification telling it that under such a condition, it is to perform such an action. Where might this specification come from?

- It might be programmed by the manufacturer, of a device, the system, or some third-party application.
- It might be programmed by the user: "Whenever I sit down, turn on the lights for me."
- It might be learned by the system, perhaps due to repetition of past direct commands by the user.

In any case, such automatic behaviors bear scrutiny and may need to be overridden; they may need to be restricted to a particular individual or a particular situation; and the user of the system might want to examine and possibly modify the behavior. How to represent, acquire, parameterize, and edit automatic behaviors is a subject that will require considerable research. At present, it is not uncommon to see demonstrations of specific devices or scenarios with specific and obvious behaviors, but it is sometimes not clear what should be generalized from these demonstrations. Also, simply making certain behaviors automatic does not make the environment "intelligent", only "somewhat automated". Intelligence requires that there be mechanisms for the growth, examination, and modification of behaviors as appropriate.

Another aspect of automatic behaviors is the desire to sometimes have them move or be copied to new locations or domains. For example, if you have a particular kind of music you like to listen to, then the associated behaviors should probably move with you, wherever you go. On the other hand, you may not want the neighbors automatically reprogramming your thermostat every time they come over for a visit. So, mechanisms may be needed to specify the scope of automatic behaviors. Adapting these behaviors may also be challenging, for example if you buy a new house, do you have to start all over again teaching it when you like to have the lights and heating turned on? And, what of the person who buys your old house?

How Do We Use The System?

Most research in intelligent environments to date centers on developing new technologies. But, just because we *can* build a complex computing system, doesn't mean we *should*. Instead, we hope to build systems that will enhance the life experiences of their users and not detract from them. This raises a number of issues:

- Who is in control? Some intelligent environments incorporate a script which they expect the user to act out, one step at a time. But, people like to interrupt, skip around, and generally be in control of the interaction. How do we enable user control over the interaction in a framework that still allows the system to conduct its business? And, what are the perceptual implications of allowing the user freedom, in effect, to act without having a restricted "grammar" of action sequences?
- Standard "widgets" have been key to the success of GUI/WIMP interfaces. What will be the widgets of successful intelligent environments and ubiquitous computing systems?
- How should the system know when the user is directing speech or gestures to the system, as opposed to others in the room, or even to nobody in particular? This can actually be very challenging, for example if you require that the user begin each command by saying "Computer, ..." or some other keyword, then what happens if the person utters that word in some other context?
- What is socially acceptable behavior for the system? For example, recording pictures of us in the shower and sending them as email to our friends would probably be considered "antisocial" if a person did it. Also if the system does it automatically! Interrupting a conversation is considered antisocial if a person does it, but we tolerate the telephone doing it all the time. Should different standards apply to people and computing systems? If a person does something we don't like, we tell them to "stop it" and expect them to, but who expects a computer to do the same? Who would even expect the computer to know what we mean by "it"? If we were to define "socially acceptable behavior", in what language could we express it to the computer?
- How do we make all of this specification manageable? Do we do it little bits at a time, over years, as we do with children? Or do we require that every conceivable situation be spelled out explicitly at once? Probably somewhere in the middle, but life is already complex, we hope we don't increase the cognitive burden too much! After all, our goal was always to make life easier, not harder.

What Makes Ubiquitous Computing Ubiquitous?

Question: How many ubiquitous computing systems will there be in the world?
Hint: How many Internets are there in the world?

Answer: Only one, if it's really ubiquitous! But it will have lots of local variation and incompatibilities and roughness from place to place. And, it will constantly be evolving.

As was true for the Internet, we can expect that the first set of issues to confront is how to get all that hardware to connect and all that software to connect. This is a fair characterization of the current efforts in mobile computing and intelligent environments. We are just beginning to see interesting examples of services emerge, but there is not yet enough standardization for them to propagate very widely in the community. Ultimately, we will need a great deal of standardization at many levels in order to achieve the degree of inter-operability that characterizes ubiquitous computing.

Who will explore, define, and ultimately control these emerging and ever-evolving standards? That is the task before us, in cooperation with the vast industry, academic, and governmental establishments invested in the future of computing.

Summary

The grand vision of ubiquitous computing raises many challenges, and requires solutions that are very broad-reaching. We have outlined a set of these challenges: meaningfulness, world modeling, user modeling, distribution, accessibility, extensibility, heterogeneity, automation, usability, and ubiquity. To date, even the most ambitious systems for mobile computing or intelligent devices are just dipping into these issues. Our challenge as a community is to reach for broader solutions to build towards that grander vision.

References

[weiser93] Weiser, M. Some Computer Science Issues in Ubiquitous Computing. *Communications of the ACM*, vol. 36 (7), July 1993, p. 75-84.

An extended version of this paper, with a more extensive bibliography, will be found on the Web at http://www.research.microsoft.com/projects/easyliving.

INTERACTION AND CONTROL

Recognition of Multi-Person Action

Aaron Bobick

College of Computing
Georgia Institute of Technology
Atlanta, GA 30332
afb@cc.gatech.edu

Abstract. Many proposed interactive systems require the ability of a machine to understand the cooperative action of several people. Unlike the activity of a single person over short time scales, multi-agent action is typically loosely defined by statistical tendencies, requires certain causal interactions, and evolves over extended periods of time. Recognition techniques designed to observe single-agent action (such as hidden Markov models) are unlikely to succeed in these situations. Here we present two approaches to the statistical recognition of multi-agent action. The first is based upon stochastic parsing of parallel event streams. This method is useful where there are a priori definitions of actions involving a small number of agents, but where the detection of individual elements is uncertain. The second approach relies on the uncertain integration of confirming evidence of large scale coordinated activity, such as a team executing a particular football play. In presentation and comparison of these two techniques we will attempt characterize multi-agent action recognition problems in terms of being structural or statistical, and in terms of their spatial-temporal rigidity.

1 Introduction

As computational hardware grows in power and shrinks in size, we begin to consider the impact of having numerous computational elements embedded in our everyday environments. Homes, offices, museums, factories, arcades are but some of the proposed domains suitable to advanced automation. Metaphors range from Negroponte's digital butler to DARPA's Command Post of the Future.

Perhaps the most fundamental requirement for any smart environment is the ability for the system to understand what the people in that environment are *doing*. The tasks may be as simple as knowing that someone is walking up the stairs to as complex as understanding the interactions of four people preparing dinner in a kitchen. Computer sensing must be able to provide sufficient information in each of these cases for the intelligent environment to act or inform in a helpful manner.

Such understanding by machine is one of the grand challenges of machine perception and in particular computer vision. While there has been a flood of work on understanding the motion of single individuals (entire conferences being dedicated to the understanding of gesture by an individual), there has been only the initial start in understanding the actions of multiple individuals interacting in a coordinated fashion.

In this paper I present two of our efforts at the machine recognition of multi-person action. Both approaches assume that such understanding requires the integration of uncertain information — sensing is always error prone and actions are rarely completely scripted. The first technique — stochastic parsing — divides the action recognition task into the two parts of primitive action detection and structural integration. The structural integration allows for requiring certain causal relations to be in effect. The second method — multi-agent action belief networks — is better suited to larger scale activity where the goal is to integrate an abundance of noisy evidence.

2 Stochastic-parsing of activities

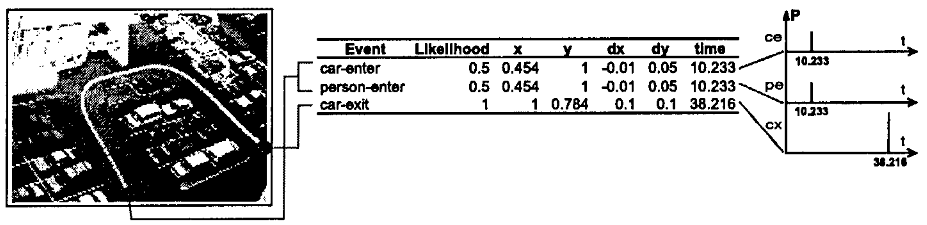
Our development of the stochastic-parsing method of activity recognition was in consideration of the general task of visual surveillance. Research in visual surveillance is quickly approaching the area of complex activities, which are framed by extended context. As more methods of identifying simple movements become available, the importance of the contextual methods increases. In such approaches, activities and movements are not only recognized at the moment of detection, but their interpretation and labeling is affected by the temporally extended context in which the events take place (e.g., see [1]). For example, when monitoring a parking lot the system observes objects of different classes — cars and people. Unfortunately, the detection of the class of an object may be uncertain. However, if when participating in an interaction the object behaves like a car, entering interactions with other objects in a way characteristic to a car, then belief about its class label is reinforced.

A complete description of the monitoring system we developed is available in [14]. Here we will briefly describe the components as each such component would be necessary for any system that uses the stochastic-parsing approach to action recognition.

We note that recently there have been several other approaches to recognizing activity that is relevant to the type of interactions described here. Some relevant work include graph interpretation models [3], non-probabilistic grammar-based parsing [2], Bayesian networks [19], and coupled hidden Markov models [17].

2.1 Activity recognition components

The activity recognition system we constructed consists of three components. The *tracker* processes the data from a camera and identifies moving objects in the camera view. The objects are tracked and the data about their movement are collected into partial tracks; details of the tracking system may be found in [20]. We now give a brief description of the *event generator* and the *parser*.



Event	Likelihood	x	y	dx	dy	time
car-enter	0.5	0.454	1	-0.01	0.05	10.233
person-enter	0.5	0.454	1	-0.01	0.05	10.233
car-exit	1	1	0.784	0.1	0.1	38.216

Fig. 1. Illustration of a process of mapping tracks onto discrete events. The tracker reports the beginning and the end of the track. In this example, the beginning of the track corresponds to an object entering the scene. At that point the class label of the class cannot be determined. This results in generation of two concurrent events - one per class (cars and persons) with probability of the label being 0.5.

Event Generator The *event generator* in our system is responsible for mapping the tracks produced by the tracker onto a set of pre-determined discrete events. These events form the basis for the syntactic constraints that the parser enforces on the input stream. The events in our system are not object-centered, but are formulated in terms of *tracker states.* While the identity of an object is not known reliably by the tracker, an object maintains the same identity throughout the detected trajectory. If the tracker can "loses" an object and then "finds" it again later due to occlusion or sudden change of lighting, the identity of the object is not preserved. Reasoning about object identity in such situations is deferred to the parser which can enforce contextual information. In this reasoning, exact configuration of the object trajectories is not important. Only the endpoints of the tracks influence how the identity is computed.

Every time the tracker reports beginning or the end of a track, the event generator produces an *event.* This set of primitive tracker states, such as `object-lost`, `object-found`, forms the alphabet of interactions.

To generate events, the event generator is given a simple *environment map.* Based on this map the event generator determines if the events occurred in "special locations", where the objects tend to enter or leave the scene. In addition, if the tracker does not have sufficient degree of confidence about the object's class, multiple candidate events are generated, one per candidate class with likelihoods assigned to each according to what's reported by the tracker. In the system there are total of 9 events, e.g. `car-enter` or `person-found`; the generation of the events is controlled by rules sensitive to factors such as where the tracks start or end, or whether the velocity dropped below a certain threshold.

The process of generating events is illustrated in figure 1. Note that some of the track endpoints are mapped onto a pair of concurrent events. In this example, when the track is started the system cannot determine if the object is a car or a person. The parser will select one or the other, depending on which labeling results in the overall parse with maximum probability. Typically, at the beginning of each track, the tracker has not observed the object long enough to be certain about its class membership. In contrast, by the time the object disappears or is lost, there is enough data to make more accurate classification decision.

Parser The *parser* analyzes the events according to a *grammar* which structurally describes possible activities. The grammar represents the knowledge about structure of possible interactions, making it possible to enforce structural and contextual constraints. Our SCFG parser is an extension of those by Earley and Stolcke (see [21]). For each input symbol, the parser keeps a *state set*, a set of parser *states* that collectively describe current pending derivations. A state is a production rule, augmented with two additional markers, i and k. Marker k indicates where in the input string the rule is applied, and marker i shows where in the rule the parser is currently located. The position of the parser inside the rule that corresponds to i is shown by the symbol ".". We denote a state as:

$$i : X_k \rightarrow \lambda . Y \mu \quad [\alpha, \gamma] \tag{1}$$

where X and Y are *non-terminals* and λ and μ are arbitrary sequences of *terminals* and *non-terminals*. In the SCFG parsing algorithm ([21]), each state also carries forward and inner probabilities denoted in (1) by α and γ, respectively. α, also called a prefix probability, is the probability of the parsed string up to position i, and γ is a probability of the sub-string starting at k and ending at i.

Parsing begins with initializing the first state set with an initial state. The parsing proceeds as an iteration between three steps - *prediction*, *scanning* and *completion* until the final state is reached. In this paper we only give a minimal exposition of the parsing algorithm, as it is presented in full elsewhere (eg. see [21] and [1].

In order to propagate track data through the parse we modify each state to include two additional auxiliary variables - $\mathbf{l}$ and $\mathbf{h}$ (a *low mark* and a *high mark* of the state). These variables hold the data about endpoints of the corresponding track:

$$\mathbf{l} = \begin{pmatrix} f_l \\ t_l \\ x_l \\ y_l \\ dx_l \\ dy_l \end{pmatrix} \qquad \mathbf{h} = \begin{pmatrix} f_h \\ t_h \\ x_h \\ y_h \\ dx_h \\ dy_h \end{pmatrix} \tag{2}$$

where f is a frame number, t - a time stamp, x and y are object coordinates in the image, and dx and dy are object velocity components.

These data are used to compute the penalty function of equation (3), which weighs total probability of the parse by joining two partial tracks with endpoints at $\mathbf{h}_1$ and $\mathbf{l}_2$. This penalty function ensures that the tracks, considered for joining, are not too far apart and are temporally consistent.

$$f(\mathbf{r}_p, \mathbf{r}_2) = \begin{cases} 0, & if \ \ (t_2 - t_1) < 0 \\ \exp\left(\frac{(\mathbf{r}_2 - \mathbf{r}_p)^T(\mathbf{r}_2 - \mathbf{r}_p)}{\theta}\right), & o/w \end{cases} \tag{3}$$

where $\mathbf{r}_p$ is computed from $\mathbf{r}_1$, based on a constant velocity assumption: $\mathbf{r}_p = \mathbf{r}_1 + d\mathbf{r}_1(t_2 - t_1)$, $\mathbf{r}_1$ and $\mathbf{r}_2$ correspond to the track endpoint positions at the track break, and $d\mathbf{r}_1$ is the instantaneous velocity of the object at position $\mathbf{r}_1$.

When an event from the event generator is received, the parser advances by one step, producing a new state set and searching it for the final state. If the final state is found, the parser traverses the parsing queue and assembles the most likely parse. After the parse is assembled, the parser outputs the resulting interpretation. Note, however, that since this operation is local, it is possible that it will be subsumed at a later time by a more general interpretation. This is the case when interpreting such interactions as `DROP-OFF` and `DRIVE-IN`, where in our grammar (shown later) the latter is a subset of the former.

In the course of developing the parser, we implemented a mechanism which allows the parser to deal with parallelism in primitives and interpretations. The latter case, parallelism in interpretations, occurs when several derivations, each related to different activities, are being traced concurrently. This form of parallelism is accounted for by traditional error recovery methods ([4]): we augment the grammar to allow arbitrary skip states so that terminals in the input stream belonging to one derivation may be skipped by a competing, overlapping derivation.

Parallelism of primitives arises becuase of concurrency: interactions that involve at least two objects, subject to their own independent consistency constraints but connected via interaction [1]. In [13] we developed a multi-class interleaved consistency filter, similar to the penalty mechanism described above, but across object classes.

Experimental Results Here we show results of the system run on a data collected on a parking lot at Carnegie Mellon University. The system runs in real time processing data from a live video feed or a video tape. The test data consisted of approximately 15 minutes of video, showing several high level events such as `drop-off` and `pick-up`.

The parser requires the interaction structure described to it in terms of Stochastic Context Free Grammar. A partial listing of the grammar employed by our system for the parking lot monitoring task is shown in figure 2. Labels in capitals are the *non-terminals* while the *terminals*, or primitives, are written in small letters. Square brackets enclose probabilities associated with each production rule. These probabilities reflect the typicality of the corresponding production rule and the sequence of primitives, which it represents.

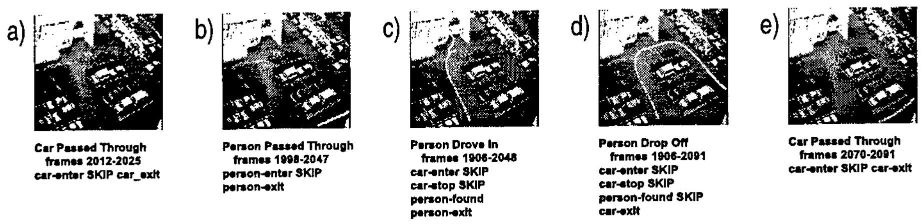
The production rule probabilities have been manually set to plausible values for this domain. Learning these probabilities is an interesting problem, which is planned for future work. However, our observations showed that the grammatical and spatial consistency requirements eliminate the majority of incorrect interpretations. This results in our system being quite insensitive to the precise values of these probabilities.

In figures 3 (a)–(e) we show a sequence of 5 consecutive detections of high level events. The sequence shown in the figure demonstrates the capability of the

```
TRACK:         CAR-TRACK              [0.5]         B-CAR-EXIT:    CAR-EXIT               [0.5]
             | PERSON-TRACK           [0.5]                      | CAR-HIDDEN CAR-EXIT    [0.5]

CAR-TRACK:     CAR-THROUGH            [0.25]        CAR-EXIT:      car-exit               [0.7]
             | CAR-PICKUP             [0.25]                     | SKIP car-exit          [0.3]
             | CAR-OUT                [0.25]
             | CAR-DROP               [0.25]        CAR-LOST:      car-lost               [0.7]
                                                                 | SKIP car-lost          [0.3]
CAR-PICKUP: ENTER-CAR-B CAR-STOP
            PERSON-LOST B-CAR-EXIT    [1.0]         CAR-STOP:      car-stop               [0.7]
                                                                 | SKIP car-stop          [0.3]
ENTER-CAR-B:    CAR-ENTER             [0.5]
              | CAR-ENTER CAR-HIDDEN  [0.5]         PERSON-LOST:   person-lost            [0.7]
                                                                 | SKIP person-lost       [0.3]
CAR-HIDDEN:     CAR-LOST CAR-FOUND    [0.5]
              | CAR-LOST CAR-FOUND
                CAR-HIDDEN            [0.5]
```

Fig. 2. A CAR-PICKUP branch of a simplified grammar describing interactions in a parking lot.



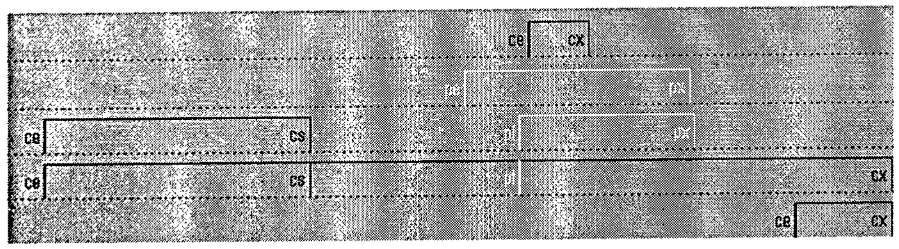
f)


Fig. 3. (a)–(e) A car passed through the scene, while DROP-OFF was performed; simultaneously a person and two other cars pass by. In (d) the car leaves the scene and the conditions for DROP-OFF are now satisfied and the label is emitted. f) Temporal extent of the actions shown in (a)–(e). Actions related to people are shown in white. Top line of the picture corresponds to the label in (a), the bottom one, (e). Car primitives are drawn in black. The figure clearly demonstrates concurrency of events. In this figure, primitive events are abbreviated as follows: **ce** - **car-enter**, **cs** - **car-stop**, **cx** - **car-exit**, **pe** - **person-enter**, **pf** - **person-found**, **px** - **person-exit**.

system to parse concurrent activities and interactions. The main event in this sequence is the DROP-OFF. While monitoring this activity, the system also detected unrelated high level events: 2 instances of CAR-THROUGH and a PERSON-THROUGH event. The figure 3f shows the temporal extent of activities, shown iconically in figures 3(a)–(e). The figure also illustrates the selectivity of the parser granted by the use of SKIP productions. Having found the interpretation PERSON-THROUGH (figure 3b) consisting of events shown in the second line of figure 3f, the parser consumes the events car-enter, person-found and car-exit by the SKIP rule.

3 Probabilistic multi-agent action recognition

The stochastic-parsing method described above is appropriate when there are causal definitions of actions that can be verified from the data. To be a `DROP-OFF` the car must enter, it must stop, a person must get out, the car must drive away. The interpretation is uncertain only because the detection of the primitives is uncertain. And the grammar is manageable because the number of primitives and agents is limited.

Such an approach is probably not ideal when we wish to recognize complex multi-agent probabilistic actions. By *complex* we simply mean that the action contains many components that occur in, typically, a partially ordered temporal relation to one another, subject to certain logical constraints (e.g. `A` happens `before` `B`, `B` is `before` `C` or `D`, but only one of `C` or `D` can occur). These relations generally reflect causal connections or influences between components. The actions we are considering are *multi-agent*, resulting in parallel event streams that interact in interesting temporal (typically causal) ways.

By *probabilistic* we refer to the uncertain nature of both the model and the data. The action description itself is typically probabilistic: e.g. `B` follows `A`, but only 80% of the time. This uncertainty results from complex actions defined by typical components that are only sometimes observed due to uncertainty in the world. Another source of uncertainty is the fuzziness of attributes used to describe agent interaction (e.g. `obj1` is `near` `obj2`). Finally, the design of the representation is intended to support recognition and we therefore need to consider real sensing capabilities, which are probabilistic at best. Often, perceptual evidence can be either missed or hallucinated.

There are numerous domains that contain interesting, complex, probabilistic actions. Examples include sporting events, military and security surveillance, traffic monitoring, and robotic collaboration. The task and domain described here is recognizing American football plays. It has the necessary attributes of containing complex actions (plays) performed by a multi-agent system (the offense) in which there is great uncertainty and unpredictability (the defense). Methods exist for tracking football players from video [11]. For the recognition task, we presume tracked data that provides the location and rough orientation of each player at each time during the play. Our current system uses a database of 29 manually, though noisily, tracked plays. Figure 4 shows two "chalkboard" image examples of two different observations of a "p51curl" play.

At the heart of our approach to complex action recognition is an idea developed within the context of model-based object recognition. Grimson and Lozano-Perez [7] noted that to recognize a complex object it was sufficient to verify numerous low-order properties such as distance and angle between pairs of edges. Global checking was not necessary if given enough local constraint. We apply the same principle in space-time: *massive low order consistency of pairwise relations typically implies correctness of interpretation.* In this section of the paper we present a brief description of our belief-net approach to the recognition of complex, multi-agent action. Details may be found in [12].

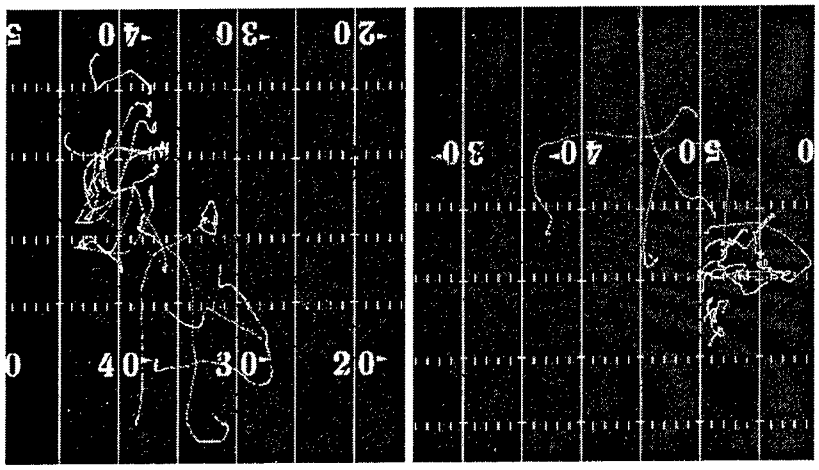


Fig. 4. Two examples of a p51curl play. The lighter trajectories are the offensive players. The data provided to the system consists of trajectories for all the objects including the ball, the approximate orientation of each object at each point along its trajectory, and a position label for each trajectory.

3.1 Our approach

The approach we have developed consists of the following representational elements:

- We first define a *temporal structure description* of the global behavior, in this case a football play. The basic elements of this structure represent individual, local goals or events that must be detected. The relations coded in the structure are temporal constraints to be verified.
- For each basic element of the temporal structure, we define a *visual network* that detects the occurrence of the individual goal or event at a given time accounting for uncertain information.
- Temporal analysis functions are defined which evaluate the validity of a particular temporal relationships, such as `before`.
- A large multi-agent belief network is automatically constructed reflecting the temporal structure of the action. This network, similar in structure to a naive Bayesian classifier, represents a particular play using *only* beliefs and evidence about the expected temporal relationships between agent goals.

The likelihood that a particular play has been observed is computed by evaluating the appropriate belief networks.

Prior work We note that our approach has been influenced by previous work in traditional plan recognition (e.g. [8]), Bayes-nets approaches to plan analysis [15, 6, 18], computer vision action recognition [16, 5, 9]. Our work builds most strongly on Huber's goal recognition belief networks [10].

3.2 s51 play example

The task for a recognition system is to recognize whether a given set of trajectory inputs like those illustrated by Figure 4 corresponds to a particular type of play, such as the p51curl. Normally plays consist of 11 offensive players. A simplified example of a p51curl play, called the "s51," containing only 4 offensive players and a reduced number of actions per player will be used for illustration in this paper. The s51 chalkboard diagram is shown in Figure 5.

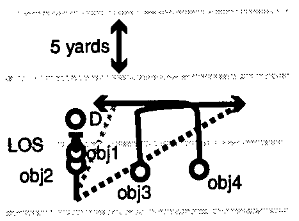


Fig. 5. An football play diagramming the s51 example play. The play consists of 4 offensive agents and a ball. Also marked is the line-of-scrimmage (LOS) and some 5-yard marker yardlines. The heavy dotted line indicates the most typical path for the ball when it is thrown by OBJ2 after the ball is handed to OBJ2 from OBJ1. The lighter dotted line indicates a secondary pass option. Implicit is that OBJ3 and OBJ4 turn at the same time.

The input to the system consists of trajectories given by (x,y,orientation,label) tuples as a function of the frame number, i.e. time. Here, *orientation* denotes the approximate upper-body orientation of the player and *label* is the name of the player's starting position.

The remainder of this section describes some elements of each component of our representation and some recognition results.

3.3 Temporal structure description

The temporal structure description represents the prototypical scenario of the described action. It is comprised of fundamental behavior elements connected by temporal constraints. We assume that the complex actions we wish to recognize have such a prototype and that they can be expressed with this language.

We use individual agent *goals* as the basis for the descriptive structure and view complex actions as a partially ordered set of goal directed behaviors on the part of interacting agents. We *define* goals by their (probabilistic) characteristic behaviors, building on work in probabilistic plan recognition [6]. To evaluate whether an agent has a particular goal at a particular time we will evaluate the perceptual evidence.

Figure 6 shows a simplified temporal structure description for the s51 example in Figure 5. The description contains four agents: obj1, obj2, obj3, and obj4.

```
(goalTeam s51
 "Team goal for simple-p51curl (s51) play."

  (agentGoal obj1
     (agent (obj1 (C))) ; Obj1 is always the Center (C)
     (goal obj1_act1 "snapToQB (obj1)")
     (goal obj2_act2 "blockQBPass (obj1)")
     (before obj1_act1 obj1_act2))

  (agentGoal obj3 ;The Right Wing Back (RWB)
     (agent (obj3 (RWB RTE RHB HB FB TB LWB LSB)))
     (goal obj3_act1 "passPatStreaking
              (obj3 4 45 defReg nearRightSidelineReg 0)")
     (goal obj3_act2 "passPatCutting (obj3 70 offSidelineRightReg
                                   freeBlockingZoneReg)")
     (goal obj3_act3 "runbehind (obj3 obj4)")
     (goal obj3_act4 "passPatParaLos
              (obj3 3 defReg offSidelineRightReg 4)")
     (goal obj3_act5 "catchPass (obj3)")
     (before obj3_act1 obj3_act2)
     (before obj3_act2 obj3_act4))

  (agentGoal obj2
     (agent (obj2 (QB))) ;Obj2 is always the Quarterback (QB)
     (goal obj1_act1 "dropback (obj2 5)")
     (goal obj2_act2 "throwPass (obj2)")
     (before obj2_act1 obj2_act2))

  (agentGoal obj4 ;The Right Flanker (RFL)
     (agent (obj4 (RFL RWB RSB LFL LSB LWB)))
     (goal obj4_act1 "passPatStreaking
                 (obj4 4 50 defReg offEndZoneReg 0)")
     (goal obj4_act2 "passPatCutting (obj4 70 offSidelineLeftReg
                                   freeBlockingZoneReg)")
     (goal obj4_act3 "passPatParaLos
                      (obj4 3 defReg offCenterLineReg 4)")
     (goal obj4_act4 "catchPass (obj4)")
     (before obj4_act1 obj4_act2)
     (before obj4_act2 obj4_act3))

  (around obj3_act2 obj4_act2)
  (xor obj3_act5 obj4_act4))
```

Fig. 6. A temporal structure description for the *s51* play example with only some actions and temporal relationships specified.

Each object in the temporal structure graph has a set of goal action components. The example indicates that in an s51 play, obj1 should have a goal to snapToQB (snap (or hand) the ball to the quarterback) and blockQBPass (block for the QB as the QB passes the ball). Each goal has a label, such as obj1_act1 (short for object1's action1). The s51 example has been limited to just six goal types: snapToQB, blockQBPass, passPatStreaking, passPatCutting, passPatParaLos, and catchPass.

Temporal constraints The remaining slots in the the temporal structure description indicate the temporal and logical relationships between agent goals. Two temporal primitives are available: *before* and *around.* For example, "(before obj1_act1 obj1_act2)" indicates that goal obj1_act1 occurs before obj1_act2, where obj1_act1 is the label for "snapToQB (obj1)" and obj2_act2 is the label for "blockQBPass (obj1)". Similarly, "(around obj3_act2 obj4_act2)" indicates that object3's passPatCutting goal occurs around the same time as object4's passPatCutting goal. The meanings of "before" and "around" will be defined shortly but we note that they are meant to be "fuzzy" and to not support transitive closure. Finally, "(xor obj3_act5 obj4_act4)" indicates that object3's catchPass goal xor object4's catchPass goal should be observed.

3.4 Visual nets and temporal functions

Previous work has shown that agent goals can be represented in a probabilistic framework using Bayesian belief networks [6, 10, 18]. We also use belief networks based on visual evidence, or *visual networks*, that offer a rich representation designed to handle uncertainty in evidence, goal models, spatial reasoning, and temporal reasoning. Further, the networks can be used as building blocks for recognizing multi-agent activity.

Figure 7 shows one such network, catchPass. The network consists of two types of nodes. **Unobservable belief nodes** have two states, *true* and *false*,

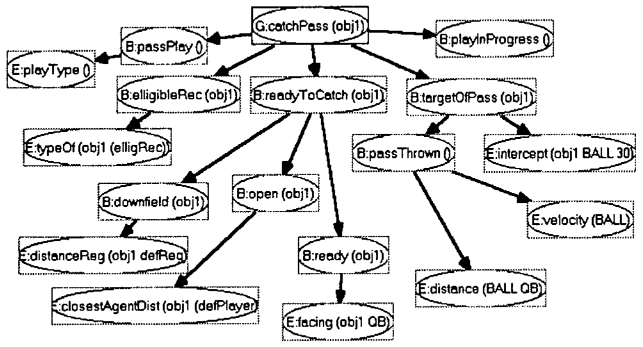


Fig. 7. The `catchPass` goal network.

and represent the internal state of the agent or some external state in the world at the time when the network is evaluated. Each visual network has a designated *main goal node* (e.g. `catchPass`). **Observable evidence nodes** states and state values are directly dependent upon the data. Some nodes are binary (e.g. *observed*, *notObserved*), most are trinary, (e.g. *observed*, *maybeObserved*, *notObserved*), and the remainder have specialized states that quantize a particular feature detector output (e.g. the result of the `distance` detector is quantized into states *inContact*, *nextTo*, *near*, *inVicinity*, *far*, *distant*).

Temporal analysis functions The output of a visual goal network at each frame for a given object results in a likelihood curve over time. Temporal relationship evidence detectors use these curves as input. The functions compute a certainty value for the *observed*, *before*, and *around* tests at each time frame using heuristic functions that compare the activation levels of each goal over time, characteristics of each input curve, the temporal distance between features of the curves, the amount of overlap between the curves, and a minimal activation time for each goal. The functions are designed to preserve the uncertainty in the output of the visual goal networks and to avoid hard thresholding. Two curves returned by the networks "dropback (QB 5)" and "catchPass (RSE)" are shown in Figure 8 overlaid with the likelihood values for the *before* and *around* detectors corresponding to "dropback (QB 5) before catchPass (RSE)" and "dropback (QB 5) around catchPass (RSE)".

3.5 Multi-agent networks

Multi-agent action is recognized using a multi-agent belief network. At each time, the network integrates the likelihood values returned by temporal analysis functions at that time and returns a likelihood that a given play has been observed.

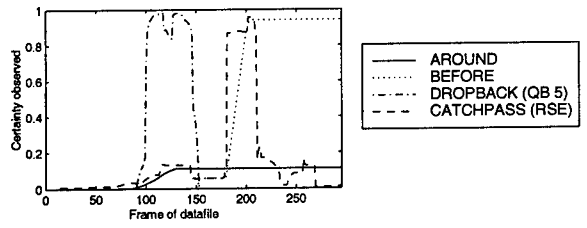


Fig. 8. Goal likelihood curves returned by the networks "dropback (QB 5)" and "catchPass (RSE)" superimposed with the corresponding temporal curves for "dropback (QB 5) before catchPass (RSE)" and "dropback (QB 5) around catchPass (RSE)".

Figure 9 shows an example of a multi-agent network for the s51 play. The network structure is generated *automatically* from the temporal structure description. In the system discussed in this paper, a two-level naive Bayesian classifier network structure is generated that encodes the temporal structure of a play. All nodes in the multi-agent networks represent beliefs or evidence observed over all the play data seen from the start of the play until the current time. The state characterization of all nodes comprises the values (*observed*, *notObserved*). The main node in the example is *B: s51 (obj1 obj2 obj3 obj4)*. Linked to that node is one node for each agent – for example *B: s51 (obj1)* – representing the belief that the agent's goals for the s51 have been observed. Below these nodes are (1) belief nodes representing binary temporal relationships between goals (e.g. *B: obj1_act1 before obj1_act2*) that reflect certainty as to whether the temporal ordering has been observed; and (2) evidence nodes for binary temporal relationships (e.g *E: obj1_act1 before obj1_act2*). To avoid cluttering the figure, these nodes are represented with a boxed "E" node.

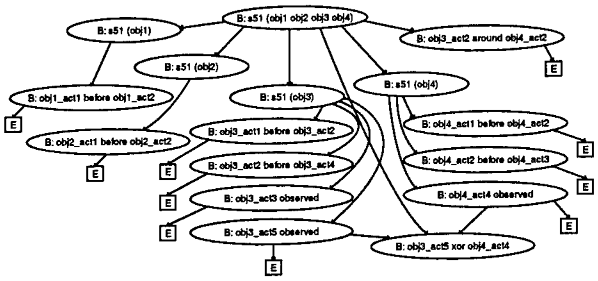

B: s51 (obj1 obj2 obj3 obj4)
B: s51 (obj1)
B: obj3_act2 around obj4_act2
B: s51 (obj2)
B: s51 (obj3)
B: s51 (obj4)
B: obj1_act1 before obj1_act2
B: obj2_act1 before obj2_act2
B: obj3_act1 before obj3_act2
B: obj3_act2 before obj3_act4
B: obj3_act3 observed
B: obj3_act5 observed
B: obj4_act1 before obj4_act2
B: obj4_act2 before obj4_act3
B: obj4_act4 observed
B: obj3_act5 xor obj4_act4
E

Fig. 9. The s51 multi-agent recognition network.

Conditional and prior probabilities for the network are determined automatically using heuristics matching table templates to specific node-link combinations, similar to the method used by Huber [10]. The structure of the network for the s51 shown in Figure 9 essentially implements a weighted voting scheme between observed goals and temporal relationships between goals.

The network shown in Figure 9 is only for a play with four agents where the number of actions for each agent is restricted to just a few examples. For a play with 11 agents, the networks typically contain *at least* 50 belief nodes and 40 evidence nodes and often twice that number. Network propagation by exact algorithms is feasible, however, because the network has a shallow tree linking structure and consists of binary internal belief nodes. The temporal analysis functions return continuous valued likelihood information. This information is entered into the multi-play network as continuous evidence, avoiding unnecessary thresholding of uncertain information.

3.6 Recognition results for football plays

We have evaluated our system on 29 tracked plays using a database of 10 temporal play descriptions. Figure 10 shows the likelihood value obtained by evaluating the multi-agent network at each frame for 7 play models on a datafile for a t39 play. Here the desired behavior is achieved: uncertain evidence of temporal relationships between goals is sufficient to cause the t39 play detector's likelihood value to quickly rise above the other plays shortly after the play action begins at frame 90.

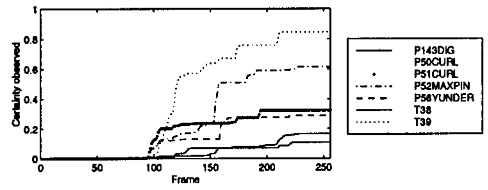


Fig. 10. Result of running 7 play detectors on a t39 play example. Shown is the likelihood of each play having been observed at frame t considering all evidence from frames $0 - t$.

Figure 11 is a confusion matrix showing the final likelihood value obtained for each temporal play description when run on 29 example plays. A "-" value indicates a play where no good object-to-trajectory consistency match could be found.[1]

The maximum likelihood value along each row selects the correct play for 21 of the 25 play instances. 3 of the 4 errors are caused by p56yunder examples being

[1] Prior to evaluating a particular multi-agent network, a consistent match between the labeled trajectories and the object label preference orderings must be found. This component of the system is not discussed in this paper.

Name	p143dig	p50curl	p51curl	p52maxpin	p54maxcross	p56yunder	p63up	p63upa	t38	t39
p143dig (file aa00185)	**.75**	.49	-	-	.37	.33	-	.24	.53	-
p143dig (file aa00412)	**.98**	.63	-	-	.75	.71	-	.57	.65	-
p143dig (file aa00606)	**.93**	.45	-	-	.57	.63	-	.32	.39	-
p143dig (file aa00847)	**.87**	.35	-	-	.53	.49	-	.27	.30	-
p143dig (file aa01032)	**.91**	.42	-	-	.50	.36	-	.60	.41	-
p143dig (file aa02128)	**.86**	.42	-	-	.43	.41	-	.70	.43	-
p143dig (file aa02329)	**.98**	.58	-	-	.85	.65	-	.57	.36	-
p50curl (file aa06046)	.19	**.87**	-	-	-	.44	-	.62	.58	.27
p51curl (file aa10542)	-	.21	**.69**	-	-	.27	.35	.34	-	.58
p51curl (file aa10736)	-	.54	**.95**	-	-	-	-	.55	-	.66
p51curl (file aa11033)	-	-	**.98**	-	-	-	-	.82	.09	.68
p52maxpin (file aa14122)	-	-	.37	**.93**	-	.66	.88	-	-	-
p54maxcross (file aa19487)	.55	.55	.37	.57	**.97**	.48	-	.77	-	-
p56yunder (file aa28294)	-	-	.47	-	-	**.63**	-	-	-	-
p56yunder (file aa29325)	-	-	.24	.51	-	**.69**	.39	-	-	-
p56yunder (file aa29486)	-	-	.75	**.88**	-	.83	.72	-	-	-
p56yunder (file aa30045)	.61	.26	-	-	-	**.80**	-	.73	.41	.47
p56yunder (file aa30560)	.38	-	.38	**.78**	-	.62	.57	-	-	-
p56yunder (file aa30761)	-	-	.54	**.76**	-	.64	.34	-	-	-
p63up (file ab00958)	-	.41	.56	-	-	-	**.87**	-	-	-
p63up (file ab01196)	-	.61	.79	-	-	-	**.95**	-	-	-
p63up (file ab01570)	-	.35	.43	-	-	-	**.89**	-	-	-
p63upa (file ab00636)	-	-	.52	-	-	-	-	.73	.12	**.76**
t38 (file bb23079)	-	-	-	-	-	-	-	.27	**.83**	.51
t39 (file bb31597)	-	.25	.39	-	-	.27	.55	.30	-	**.83**

Fig. 11. Likelihood values when the recognition system is run for each of 10 play models over a dataset of 29 examples.

misclassified as p52maxpin plays. In actuality these two plays are identical except for optional actions executed by two players. Without greater play detail and without all the necessary visual networks, we cannot expect good discriminability between these two plays.

4 Multi-person actions: rules or tendencies?

We have outlined two of our approaches to the recognition of multi-person action. While both are probabilistic, they are quite different in their view of how an action is defined. The stochastic-parsing approach presumes that the actions of interest have definitions or conditions that are necessary and/or sufficient The multi-agent network technique, however, views actions as statistical tendencies. Which is better for smart environments? For places with crowds, e.g. museums, the statistical approach seem the only possibility. For places with defined tasks — a hospital alert system that notices if no nurse is within hearing distance of an intercom — the ability to incorporate rules within a probabilistic framework seems advantageous. But for the truly smart environments — a home that allows an elderly person to live at home years longer than she would otherwise — we do not yet know. It will depend on what we need to see.

Acknowledgements I am deeply indebted to Yuri Ivanov and Stephen Intille who did all the hard work reported here and who co-authored some of the papers from which some of this material has been gathered.

References

1. A. Bobick and Y. Ivanov. Action recognition using probabilistic parsing. In *Proc. Comp. Vis. and Pattern Rec.*, pages 196–202, Santa Barbara, CA, 1998.
2. M. Brand. Understanding manipulation in video. In *AFGR96*, pages 94–99. 1996.
3. Frank Z. Brill, Thomas J. Olson, and Christopher Tserng. Event recognition and reliability improvements for the autonomous video surveillance system. In *Image Understanding Workshop*, pages 267–283, Monterey, CA, 1998.
4. H. Bunke and D. Pasche. Parsing multivalued strings and its application to image and waveform recognition. *Structural Pattern Analisys*, 1990.
5. H. Buxton and S. Gong. Advanced visual surveillance using Bayesian networks. In *Proc. of the Workshop on Context-Based Vision*, pages 111–123, Cambridge, MA, June 1995. IEEE Computer Society Press.
6. E. Charniak and R. P. Goldman. A Bayesian model of plan recognition. *Artificial Intelligence*, 64:53–79, 1993.
7. W.E.L. Grimson and T. Lozano-Pérez. Localizing overlapping parts by searching the interpretation tree. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 9(4):469–482, 1987.
8. B.J. Grosz and S. Kraus. Collaborative plans for complex group action. *Artificial Intelligence*, 86(2):269–357, 1996.
9. T. Huang, D. Koller, J. Malik, G. Ogasawara, B. Rao, S. Russell, and J. Weber. Automatic symbolic traffic scene analysis using belief networks. In *Proc. Nat. Conf. on Artificial Intelligence*, pages 966–972. AAAI Press, 1994.
10. M.J. Huber. *Plan-based plan recognition models for the effective coordination of agents through observation.* PhD thesis, University of Michigan, 1996.
11. S.S. Intille and A.F. Bobick. Closed-world tracking. In *Proceedings of the Fifth International Conference on Computer Vision*, pages 672–678, Los Alamitos, June 1995.
12. S.S. Intille and A.F. Bobick. Representation and visual recognition of complex, multi-agent actions using belief networks. In *Proceedings of the ECCV 98 Workshop on Perception of Human Action*, June 1998.
13. Yuri Ivanov and Aaron Bobick. Parsing multi-agent interactions. Technical Report 479, MIT Media Lab, December 1998 1998.
14. Y. Ivsnov and A Bobick. Recognition of multi-agent interaction in video surveillance. In *Int Conf. Comp. Vis.*, pages 169–176, Corfu, 1999.
15. H.A. Kautz and J.F. Allen. Generalized plan recognition. In *Proc. Nat. Conf. on Artificial Intelligence*, pages 32–37, August 1986.
16. H.-H. Nagel, H. Kollnig, M. Haag, and H. Damm. Association of situation graphs with temporal variations in image sequences. In *Computational Models for Integrating Language and Vision*, pages 1–8, November 1995.
17. Nuria Oliver, Barbara Rosario, and Alex Pentland. Statistical modeling of human interactions. In *CVPR, The Interpretation of Visual Motion Workshop*, pages 39-46, Santa Barbara, CA, 1998. IEEE.
18. D.V. Pynadath and M.P. Wellman. Accounting for context in plan recognition with application to traffic monitoring. In P. Besnard and S. Hanks, editors, *Int'l Conference on Uncertainty in Artificial Intelligence*, volume 11, 1995.
19. P. Remagnino, T. Tan, and K. Baker. Agent orientated annotation in model based visual surveillance. In *ICCV*, pages 857–862, Bombay, 1998.
20. Chris Stauffer and W.E.L. Grimson. Adaptive background mixture models for real-time tracking. 1999.
21. A. Stolcke. An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. *Computational Linguistics*, 21(2), 1995.

Tracking Fingers and Hands with a Rigid Contour Model in an Augmented Reality

Daniela Hall and James L. Crowley

PRIMA group— Lab. GRAVIR-IMAG
INRIA Rhônes-Alpes
655, avenue de l'Europe
38330 - Montbonnot Saint Martin, France

Abstract. Within an intelligent environment, tracking of hands and fingers allows users to directly interact with information technology without using a keyboard or mouse. Observing the way that people grasp and manipulate objects allows any convenient object to serve as a numerical input device. The identity of the object can be used to indicate the command to be executed, while the manipulation actions can be used to indicate parameters for commands.
This article describes a real time finger tracker based on rigid contours. The finger tracker is precise and performs well under difficult lighting conditions. The finger tracker is robust to background clutter when used in combination with color detection. Based on the finger tracking technique, the article describes a robust hand tracking module that allows the detection of several hand configurations.
This finger tracker has been developed as part of a project to construct a "smart office" which uses projection of information and computer vision to provide new forms of access to information technology in office tasks.

1 Introduction

Intelligent environments based on ubiquitous sensing and ubiquitous display can liberate the user from interaction using peripherical tools. Such environments allow any object to take on the role of a numerical input device. When an object is grasped, its identity can be used to indicate a command to be executed. The manner in which the object is manipulated can be used to represent parameters for the command. Such functionality requires a means to track and interpret the user's hands and fingers.

Many research groups work on finger and object tracking using a wide variety of different approaches [IB98,HS95,MDC98]. Each approach is strong in one specific case. Our goal is to design a system that is capable of detecting and following a pointer instrument such as a finger, a pen, or a bar, in image sequences.

In this article we describe a technique which uses rigid contours to track finger tips. We employ this as an ensemble of five finger trackers to construct a robust hand tracker. This finger tracker is illustrated in the context of an white board augmented by a video projector and camera [Sau97,HGM+99,MIEL99]. Functions on this white board are inspired by the DigitalDesk as described in [Wel93,MVC+93,NW92].

Augmented white boards, as well as the DigitalDesk, use a flat surface as both a projection screen and a work space. Because the workspace can be cluttered, there are problems with background. Projection of an image by a video projector introduces difficult problems with lighting and video noise. Another problem occurs when these applications are used as collaborative tools. The number of pointing instruments can change and pointers can have different properties of shape, color, or reflection. In addition many applications require a high precision and robustness in order to operate.

To accommodate these constraints, we have designed a tracking system based on rigid contours within a parametric search region. The rigid contour approach is highly precise and returns the result in image coordinates that can be transformed to coordinates of the scene plane. This approach is robust to changing lighting and can be used with a variety of pointing instruments with differing shapes and colors. The number of tracked targets is limited only by the available computing power. This allows the implementation of a robust hand tracker based on independent tracking of the fingers of one hand.

The following section describes the implementation of the system. Section 3 shows performance of the system on an integrated prototype called the MagicBoard. Section 4 applies the rigid contour in order to obtain an robust hand tracking tool, that uses the redundancy of the strongly correlated relative position of fingers of a single hand. In section 5 the characteristics of the tracker are discussed.

2 Finger and Pen Tracking

Object tracking in a video sequence can be divided into subproblems, which correspond to the search of position and orientation of the object in the next frame I_t. The knowledge of previous tracking steps such as position and orientation in frame I_{t-1} or camera movement can be used. So tracking can be reduced to the problem of the position and orientation change within two consecutive frames. The proposed algorithm is not limited to fingers, although they are used in the examples. This algorithm can track any object, that has a distinctive shape and undergoes only a small deformation during 3D motion such as pens, pointers, or markers.

2.1 Tracking phase

The position and the orientation of the targets is determined using a *rigid contour model (RCM)* of the shape of the target. It consists of several points attached to the target contour and a center point (see figure 2(b)). These points are found by maximizing the energy of the gradient. A measurement for the presence of a target is the sum $S = \Sigma_{i=1}^{k} e(p_i)$ of the gradient energy $e(p_i)$ of all k model points. The relative position of the points describe the shape of a target tip. In the case of fingers, pens and markers, the rigidity of the shape model is not a restriction, because these objects have only a slight deformation during motion. They can therefore be described precisely enough using the RCM.

To find the position and orientation in two consecutive frames, all possible transformations of the form 1 are applied to the RCM.

$$\begin{pmatrix} p_x{}' \\ p_y{}' \\ 1 \end{pmatrix} = \begin{pmatrix} cos(\Delta\theta) & -sin(\Delta\theta) & \Delta x \\ sin(\Delta\theta) & cos(\Delta\theta) & \Delta y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} p_x \\ p_y \\ 1 \end{pmatrix} \tag{1}$$

These transformations keep the center point within a small search region (see figure 1). The transformations are determined by discrete values within a predefined parametric search region for rotation angles, $\Delta\theta$, and translations, Δx and Δy. The transformation, T, with the highest score, S, is the transformation that fits the RCM best to the target in frame I_t. This transformation approximates the movement of the target from $t-1$ to t. In our experiment we use the above transformations. The parametric search regions can be adapted to the required precision and speed of the application. If required these transformations can be extended to projective transformations that are capable to track the finger even when it undergoes important 3D motion.

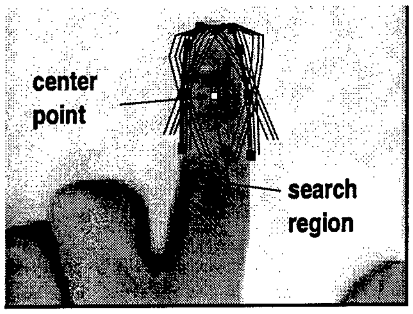


Fig. 1. A few transformed RCMs that are checked during the process. The center point is kept within the search region.

The spiral search is used to speed up the search. The finger position can be predicted using the position of the previous frame, kalman filtering or the global hand model. The parameters $\Delta\theta$, Δx and Δy for the transformation are ordered by increasing distance from the predicted finger position. The search continues until a transformation with a sufficient high score, S, is obtained. This method has the advantage that not every possible transformation of the search region needs to be performed.

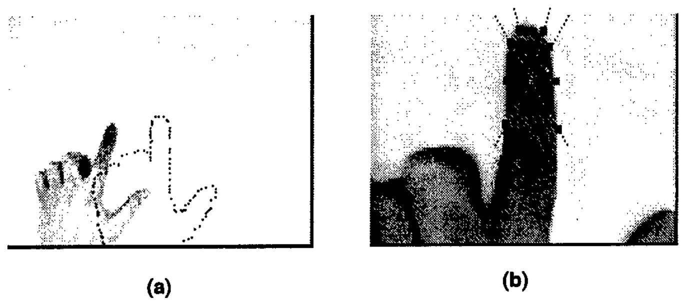


Fig. 2. (a) Initialization by projecting contour of hand at the initialization position. (b) Initialization algorithm.

2.2 Initialization phase

Precise initialisation is essential for reliable tracking. Both interactive and automatic initialisation are implemented in the current version of the system. Interactive initalisation has the advantage that several targets can be initialised, that do not have any geometric relation (such as pointer instruments of independent collaborators). The automatic initialisation has the advantage that no outer assistance is needed.

Interactive Initialisation. The user places his hand in the region that is observed by the camera. The grabbed image is displayed on the screen. The finger position is determined by selecting a point in the middle of the finger tip and a second that specifies together with the first point the direction orthogonal to the direction of the finger. The direction orthogonal to the finger is preferred because the shorter distance produces smaller errors. The other points attached to the contour are then computed relative to the two initialisation points. The points are placed at the position with the highest gradient along a ray starting at the center point. Figure 2 (b) illustrated this algorithm. This technique has the advantage that the RCM can be attached to any object. Successful tracking is guaranteed with objects that have a small deformation during 3D motion and form a visible contour against the background.

Automatic Initialisation. The second possible method is to mark the hot spot region on the scene by projecting an image of a hand using the feed back medium. The user is then invited to place his hand on the projected hand model (see figure 2(a)). From the view point of the user, this action is very natural and no additional operation binds the user to a physical workstation. From the point of view of the system, the projection of the shape of a hand is an easy way to indicate the hot spot in which the initialization can take place.

The initialization loop is started by displaying the contour of a hand and stopped when the initialization has either been successful or has failed a maximum number of attempts. The initialization algorithm is based on the algorithm used for the interactive initialisation. The two points needed by the algorithm are fixed by the displayed contour. To avoid contours that soon would fail, the sum S of the gradient and the overall width and height of the contour are considered.

This method has been found well suited for an application such as the MagicBoard because no other interaction than putting the hand at the spot indicated by the contour is required. Another advantage is that the hand can be displayed in case of a complete target loss. This allows quick re–initialization during runtime.

The disadvantage of the RCM approach is the sensitivity to background clutter with a high gradient. During the initialisation process this clutter can produce an intialisation of points that are not part of the finger contour. The resulting shape does therefore not correspond to the shape of a finger tip and the tracking process will fail in the first step after the initialisation. For a correct initialisation an uniform background is indispensable.

2.3 Combination with object detection using color

The application environment contains a cluttered background with writing or drawings. In some cases, the gradient energy of such background can be greater than the gradient introduced by the finger contour. The combination of such high gradient clutter and sudden movements of the target can cause a target loss and a time-expensive reinitialisation process.

Frequent reinitialisation disturbs the user and decreases the over all performance of the system. Thus we have experimented with the use of other methods to reinforce the reliability of tracking. One such method is skin color detection, as described in [SW95]. The combination of color detection with the contour method promises to increase the robustness of the system and reduce the number of target losses due to background clutter and sudden movements.

During the initialisation process (section 2.2) a color histogram is initialised by using the region within the contour as color sample. During run time the pixels within the RCM are tested for correct color. To save computing costs only the pixels on the line between the first and the last point are considered for the color test. This test provides a supplemental constraint that diminishes target losses on cluttered background.

3 Performance

To measure the performance of the tracker, a test person was asked to follow a line of a certain length with constant speed. The target has been initialised using 8 contour points. Each experiment contains between 40 and 80 repetition for each parameter group. The test person had some difficulties to move the finger with constant speed. That means the measured percentages have variances in acceleration as they occur in many application.

Speed [cm/s]	8	10	12	15	frequency [Hz]
1 target	-	0.0%	3.8%	13.2%	16.9
2 targets	0.0%	9.7%	15.4%	-	7.6

Table 1. Percentage of target losses during experiment.

The experiment shows that the number of tracked targets is an important parameter, because it affects the frequency and the working speed (speed at which fewer than 3% errors occur). All target losses, that occurred, are due to confusions with another finger as shown in figure 3. There are a lot of ambiguities in the shape of the fingers of a hand. Confusion occurs when the distance from the predicted position of another finger with ambiguous shape is smaller than the distance of the target finger. In this case the wrong finger will be tracked. It should be noted that during the experiment the hand was never lost. This allows reinitialisation to take place in a relatively small region of interest.

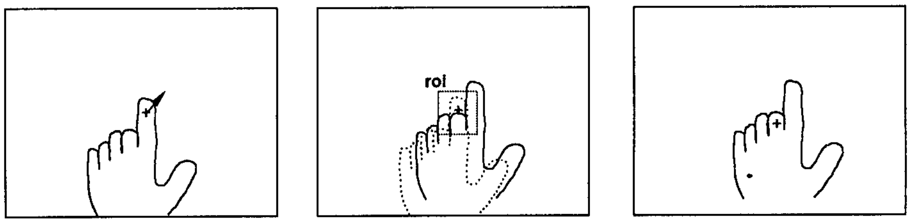


Fig. 3. Tracking error due to confusion between different fingers.

In the experiment, one pixel corresponds to 1.9mm. This implies a working speed for two fingers of about 9cm/s and for one finger of about 11cm/s. This corresponds to a relaxed drawing speed. This speed is sufficiently fast for most interactions, but can be sometimes be exceeded by sudden accelerations of the hand. Position accuracy is limited by the resolution of the camera. It is within ± 1 pixel ($\approx$ 1.9mm) and is sufficiently precise for projected applications because they are magnified.

The 2D pointing direction of the finger is obtained as a side effect, which can be useful for pointing or modeling in the case of several fingers. It can also be used to detect gross tracking errors using previous knowledge about unlikely target orientations in the setup. The number of targets that are tracked simultaneously is not limited and computation time increases linearly with the number of targets.

The tracking module is designed to permit using a finger as a pointing device in the same style as a mouse in a desktop environment. A mouse-down event can be detected in several ways: listening for tapping noise on the board, immobility for a specified time, making specified gesture or performing a movement of thumb in direction of index finger in the case of tracking of two targets. We have experimented with these last two possibilities and we are configuring the system to listen to tapping noise.

The preferences of the users vary. The tracking of two targets lacks speed and is harder to initialise. The performance of a specified thumb gesture makes involuntary clicks impossible but a tiredness can occur after a period of time. The technique of a certain time of immobility has the advantage of easier reinitialisation and overall faster tracking but involuntary click can occur during a reflection break of the user.

4 Robust hand tracker

Hands have very strong constraints concerning the relative position of the different fingers. Figure 4 illustrates these constraints. The circles that center the finger tips with a constant radius for each finger intersect in a single point, the physical rotation point of the fingers. This remains true when the configuration of the hand changes within the observation plane. When the camera zoom changes or the distance hand camera is changed the same radii can be used, because the circles still intersect in a single point.

This constraint can now be used to build a robust hand tracking module. We assume that the hand is defined by the position of the five finger tips, $P_i, i \in \{0,1,2,3,4\}$. The initialisation process attaches to each finger a RCM. Then a point near the wrist is selected as reference point, B^{t_0}. The distance from this reference point to each RCM is computed and stored. These are the initial radii, $r_i, i \in \{0,1,2,3,4\}$.

After each tracking step a new reference point, B^{t_k}, is computed by intersecting two circles, $C_{i_1}^{t_k} = \left(P_{i_1}^{t_k}, r_{i_1}\right), C_{i_2}^{t_k} = \left(P_{i_2}^{t_k}, r_{i_2}\right), i_1, i_2 \in \{0,1,2,3,4\}$. The difference of the distances, $d_i = \overline{P_i^{t_k} B^{t_k}}$, and the initial radii, r_i, is an error measure. Incorrect tracking results can be detected immediately.

After the error detection step the falsely tracked fingers are reinitialised using the information of the correctly tracked fingers. As few as two correctly tracked fingers are required to predict reliably the position of the other fingers when the hand is in an outstretched configuration.

A temporary angled finger does not cause problems because as soon as it switches to outstretched state it will be recaptured. The information that a finger is not present at the predicted position shows that the configuration of the hand does not belong to the group of outstretched configurations, which can be used to trigger a gesture recognition module as implemented in the MagicBoard prototype. Another advantage is that confusions between several fingers of one hand as shown in figure 3 are detected and the incorrect RCM is reinitialised on the correct target.

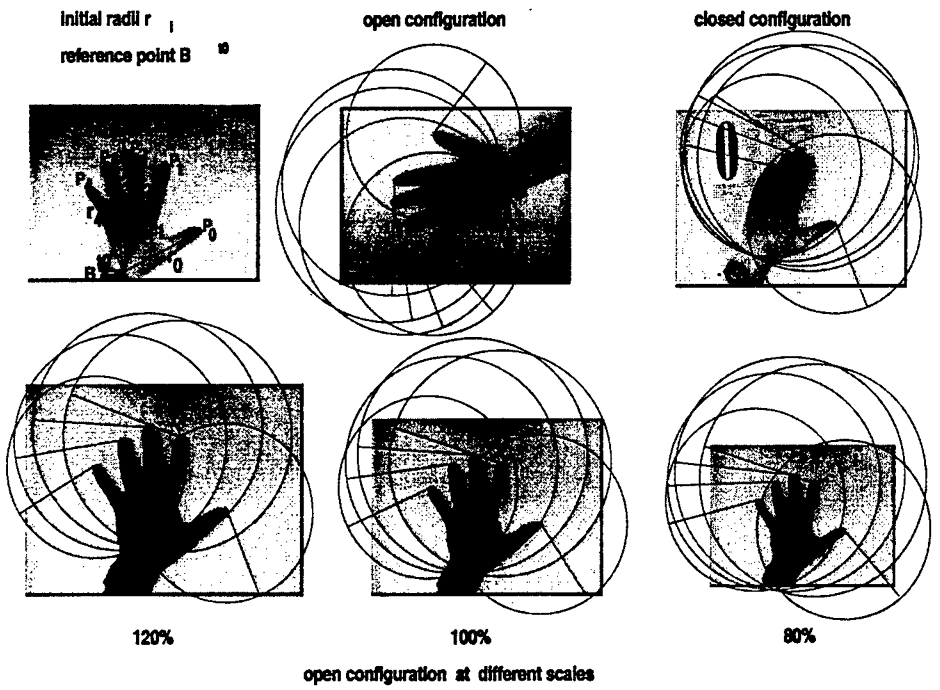


Fig. 4. Constraint of invariant distance of fingers tips P_i to reference point B. In the intialisation phase (top left), the distances, r_i of the finger tips, P, to the center point, B, are stored. Each distance represents the radius of a circle which gives a possible position for the reference point in the next image. The new reference point is determined from the intersection of the circles. This constraint is invariant against different hand configurations (open, closed) and scale.

5 Discussion

In this article we presented a system for pointer object tracking which is based on a rigid contour model. The advantage of the rigid contour approach is the high precision and the robustness to changing lighting conditions. The combination of the approach with color detection allows the operation in a sparsely cluttered background. Many different pointing instruments and other objects can be tracked without further configuration of the system. The scene is located in a known plane, which means that two dimensional tracking coordinates can be transformed into scene coordinates. The system is capable of tracking several targets independently and the computation time increases linearly with the number of targets.

There remain some problems with this approach. On a cluttered background the initalisation process might select points that do not belong to the contour of the object. This implies that not all contour points are attached to the object contour and leads to a tracking failure as soon as the target starts moving. This problem is avoided when initialisation takes place on a uniform background. The confusions between different fingers of one hand is difficult to avoid. In case of the pointing gesture as shown in figure 2(b) the addition of points to the RCM, such that the RCM reaches the joint of the middle finger, can improve the results.

Using the geometric constraints of the relative finger positions of a single hand allows the tracker to predict finger positions and to detect tracking errors which makes the system to a robust hand tracking tool. The most common tracking error as shown in figure 3 can be detected and avoided. The hand tracker can recognise all possible outstretched configurations. Lost targets imply a configuration that does not belong to the family of outstretched configurations. The disadvantage of the robust hand tracker is the fact that 5 targets need to be tracked simultaneously which implies a high load of the CPU.

References

[HGM$^+$99] D. Hall, C. Le Gal, J. Martin, O. Chomat, T. Kapuscinski, and J.L. Crowley. Magicboard: A contribution to an intelligent office environment. In *7th International Symposium on Intelligent Robotic Systems (SIRS '99)*, Coimbra, Portugal, July 1999.

[HS95] T. Heap and F. Samaria. Real-time hand tracking and gesture recognition using smart snakes. Technical report, Olivetti Research Limited, 1995.

[IB98] M. Isard and A. Blake. Condensation - conditional density propagation for visual tracking. *International Journal Computer Vision*, 1998.

[LZ97] A. Lux and B. Zoppis. An Experimental Multi-language Environment for the Development of Intelligent Robot Systems. In *5th International Symposium on Intelligent Robotic Systems, SIRS'97*, pages 169–174, 1997. more informations at `http://www-prima.imag.fr/Ravi/`.

[MDC98] J. Martin, V. Devin, and J.L. Crowley. Active hand tracking. In *IEEE Third International Conference on Automatic Face and Gesture Recognition, FG '98*, Nara, Japan, April 1998.

[MIEL99] E.D. Mynatt, T. Igarashi, W.K. Edwards, and A. LaMarca. Flatland: New dimensions in office whiteboards. In *Human Factors in Computing Systems (CHI '99)*, pages 346–353, Pittsburgh, PA, USA, May 1999.

[MVC+93] W.E. Mackay, G. Velay, K. Carter, C. Ma, and D. Pagani. Augmenting reality: Adding computational dimensions to paper. *CACM*, 36(7):96–97, 1993.

[NW92] W. Newman and P. Wellner. A desk supporting computer–based interaction with paper documents. In *ACM Conference on human factors in computing systems (CHI'92)*, pages 587–592, May 1992.

[Sau97] E. Saund. Machine perception in support of instrumented office white-boards. In *Workshop on Perceptual User Interfaces (PUI'97)*, pages 33–25, Banff, Alberta, Canada, October 1997.

[SW95] B. Schiele and A. Waibel. Estimation of the head orientation based on a face–color–intensifier. In *3rd International Symposium on Intelligent Robotic Systems '95*, July 1995.

[Wel93] P. Wellner. Interacting with paper on the digitaldesk. *CACM*, 36(7):86–95, 1993.

Experiments presented in this article were programmed using the RAVI multi–language environment [LZ97].

Combining Audio and Video in Perceptive Spaces

Christopher R. Wren, Sumit Basu, Flavia Sparacino, and Alex P. Pentland

Perceptual Computing Section
The MIT Media Laboratory
20 Ames St., Cambridge, MA 02139 USA
{cwren,flavia,sbasu,sandy}@media.mit.edu
http://www.media.mit.edu/vismod/

Abstract. Virtual environments have great potential in applications such as entertainment, animation by example, design interface, information browsing, and even expressive performance. In this paper we describe an approach to unencumbered, natural interfaces called Perceptive Spaces with a particular focus on efforts to include true multi-modal interface: interfaces that attend to both the speech and gesture of the user. The spaces are unencumbered because they utilize passive sensors that don't require special clothing and large format displays that don't isolate the user from their environment. The spaces are natural because the open environment facilitates active participation. Several applications illustrate the expressive power of this approach, as well as the challenges associated with designing these interfaces.

1 Introduction

We live in real spaces, and our most important experiences are interactions with other people. We are used to moving around rooms, working at desktops, and spatially organizing our environment. We've spent a lifetime learning to competently communicate with other people. Part of this competence undoubtedly involves assumptions about the perceptual abilities of the audience. This is the nature of people.

It follows that a natural and comfortable interface may be designed by taking advantage of these competences and expectations. Instead of strapping on alien devices and weighing ourselves down with cables and sensors, we should build remote sensing and perceptual intelligences into the environment. Instead of trying to recreate a sense of place by strapping video-phones and position/orientation sensors to our heads, we should strive to make as much of the real environment as possible responsive to our actions.

We have therefore chosen to build vision and audition systems to obtain the necessary detail of information about the user. We have specifically avoided solutions that require invasive methods, like special clothing, unnatural environments, or even radio microphones.

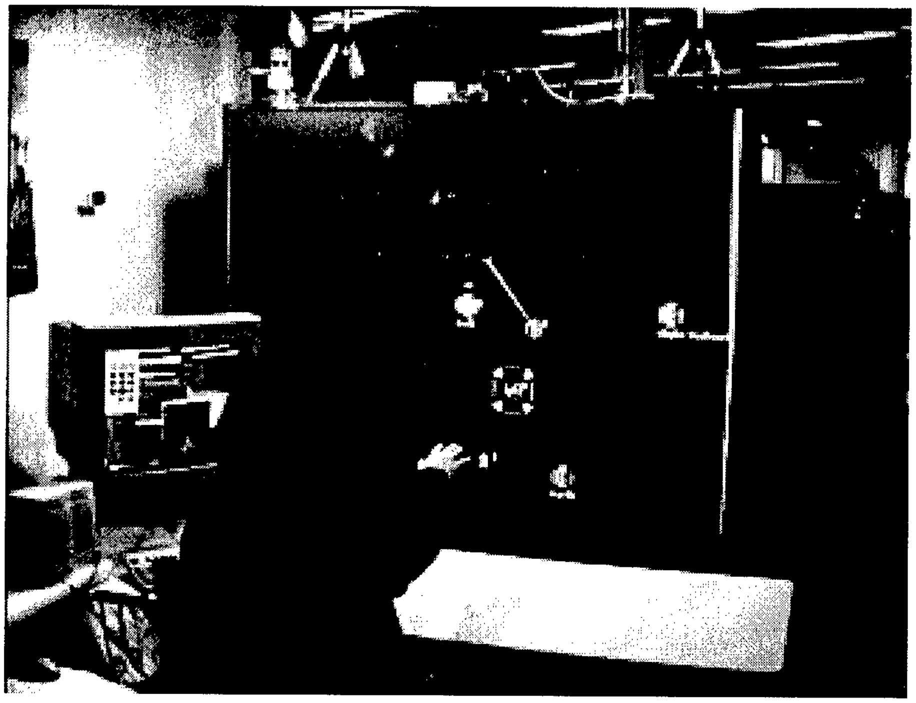


Fig. 1. Netrek Collective Interface: the user issues audio and gestural information in conjunction.

This paper describes a collection of technology and experiments aimed at investigating this domain of interactive spaces. Section 2 describes some our solutions to the non-invasive interface problem. Section 3 discusses some of the design challenges involved in applying these solutions to specific application domains with particular attention paid to the whole user: both their visual appearance, and the noises that they make.

2 Interface Technology

The ability to enter the virtual environment just by stepping into the sensing area is very important. The users do not have to spend time "suiting up," cleaning the apparatus, or untangling wires. Furthermore, social context is often important when using a virtual environment, whether it be for game playing or designing aircraft. In a head mounted display and glove environment, it is very difficult for a bystander to participate in the environment or offer advice on how to use the environment. With unencumbered interfaces, not only can the user see and hear a bystander, the bystander can easily take the user's place for a few seconds to illustrate functionality or refine the work that the original user was creating. This section describes the methods we use to create such systems.

2.1 The Interactive Space

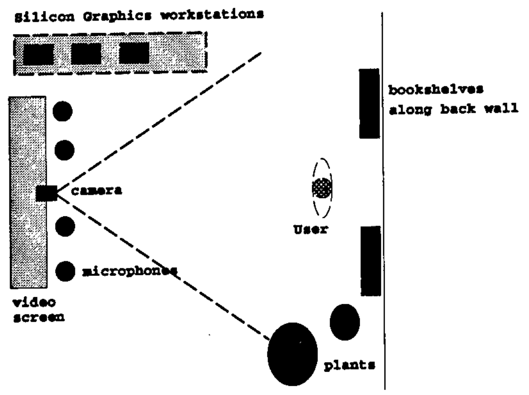


Fig. 2. Interactive Virtual Environment hardware.

Figure 6 demonstrates the basic components of an Interactive Space that occupies an entire room. We also refer to this kind of space as an Interactive Virtual Environment (IVE). The user interacts with the virtual environment in a room sized area (15'x17') whose only requirements are good, constant lighting and an unmoving background. A large projection screen (7'x10') allows the user to see the virtual environment, and a downward pointing wide-angle video camera mounted on top of the projection screen allows the system to track the user (see Section 2.2). A narrow-angle camera mounted on a pan-tilt head is also available for fine visual sensing. One or more Silicon Graphics computers are used to monitor the input devices in real-time.[10].

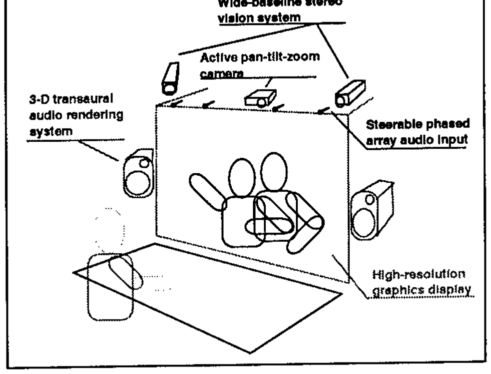

Wide-baseline stereo vision system
Active pan-tilt-zoom camera
3-D transaural audio rendering system
Steerable phased array audio input
High-resolution graphics display

Fig. 3. An Instrumented Desktop

Another kind of Interactive Space is the desktop. Our prototype desktop systems consist of a medium sized projection screen (4'x5') behind a small desk (2'x5'—See Figure 3). The space is instrumented with a wide-baseline stereo camera pair, an active camera, and a phased-array of microphones. This configuration allows the user to view virtual environments while sitting and working at a desk. Gesture and manipulation occur in the workspace defined by the screen and desktop. This sort of interactive space is better suited for detailed work.

2.2 Vision-based Blob Tracking

Applications such as unencumbered virtual reality interfaces, performance spaces, and information browsers all have in common the need to track and interpret human action. The first step in this process is identifying and tracking key features of the user's body in a robust, real-time, and non-intrusive way. We have chosen computer vision as one tool capable of solving this problem across many situations and application domains.

We have developed a real-time system called Pfinder[13] ("person finder") that substantially solves the problem for arbitrarily complex but single-person, fixed-camera situations(see Figure 4a). The system has been tested on thousands of people in several installations around the world, and has been found to perform quite reliably.[13]

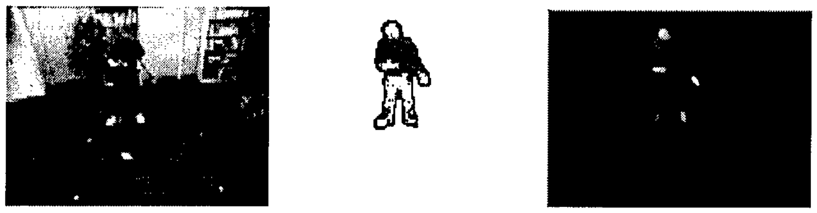


Fig. 4. Analysis of a user in the interactive space. Frame **(left)** is the video input (n.b. color image possibly shown here in greyscale for printing purposes), frame **(center)** shows the segmentation of the user into blobs, and frame **(right)** shows a 3-D model reconstructed from blob statistics alone (with contour shape ignored).

Pfinder is descended from a variety of interesting experiments in human-computer interface and computer mediated communication. Initial exploration into this space of applications was by Krueger [7], who showed that even 2-D binary vision processing of the human form can be used as an interesting interface. Pfinder goes well beyond these systems by providing a detailed level of analysis impossible with primitive binary vision.[13]

Pfinder uses a stochastic approach to detection and tracking of the human body using simple $2\frac{1}{2}$-D models. It incorporates *a priori* knowledge about people primarily to bootstrap itself and to recover from errors. This approach allows

Pfinder to robustly track the body in real-time, as required by the constraints of human interface.[13]

Pfinder provides a modular interface to client applications. Several clients can be serviced in parallel, and clients can attach and detach without affecting the underlying vision routines. Pfinder performs some detection and classification of simple static hand and body poses. If Pfinder is given a camera model, it also back-projects the 2-D image information to produce 3-D position estimates using the assumption that a planar user is standing perpendicular to a planar floor (see Figure 4c)[13].

2.3 Stereo Vision

The monocular-Pfinder approach to vision generates the $2\frac{1}{2}$-D user model discussed above. That model is sufficient for many interactive tasks. However, some tasks do require more exact knowledge of body-part positions.

Our success at 2-D tracking motivated our investigation into recovering useful 3-D geometry from such qualitative, yet reliable, feature finders. We began by addressing the basic mathematical problem of estimating 3-D geometry from blob correspondences in displaced cameras. The relevant unknown 3-D geometry includes the shapes and motion of 3-D objects, and optionally the relative orientation of the cameras and the internal camera geometries. The observations consist of the corresponding 2-D blobs, which can in general be derived from any signal-based similarity metric.[1]

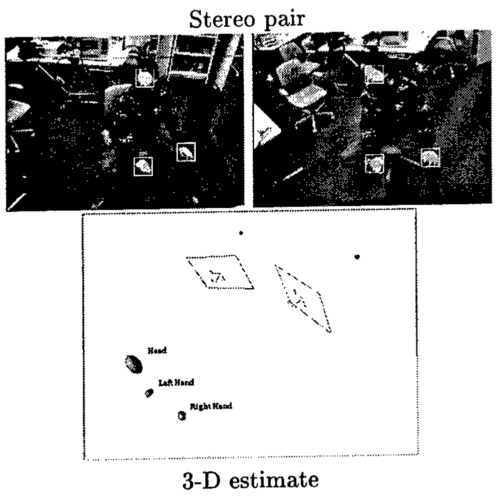


Fig. 5. Real-time estimation of position, orientation, and shape of moving human head and hands.

We use this mathematical machinery to reconstruct 3-D hand/head shape and motion in real-time (30 frames per second) on a pair of SGI O2 workstations without any special-purpose hardware.

2.4 Physics-Based Models

The fact that people are embodied places powerful constraints on their motion. An appropriate model of this embodiment allows a perceptual system to separate the necessary aspects of motion from the purposeful aspects of motion. The necessary aspects are a result of physics, are predictable. The purposeful aspects are the direct result of a person attempting to express themselves through the motion of their bodies. By taking this one thoughtful step closer to the original intentions of the user, we open the door to better interfaces. Understanding embodiment is the key to perceiving expressive motion.

We have developed a real-time, fully-dynamic, 3-D person tracking system that is able to tolerate full (temporary) occlusions and whose performance is substantially unaffected by the presence of multiple people. The system is driven by *blob features* as described above. These features are then probabilistically integrated into a fully-dynamic 3-D skeletal model, which in turn drives the 2-D feature tracking process by setting appropriate prior probabilities.

This framework has the ability to go beyond passive physics of the body by incorporating various patterns of control (which we call 'behaviors') that are *learned* from observing humans while they perform various tasks. Behaviors are defined as those aspects of the motion that cannot be explained by passive physics alone. In the untrained tracker these manifest as significant structure in the innovations process (the sequence of prediction errors). Learned models of this structure can be used to recognize and predict this purposeful aspect of human motion.[14]

2.5 Visually Guided Input Devices

Robust knowledge of body part position and body pose enables more than just gross gesture recognition. It provides boot-strapping information for other methods to determine more detailed information about the user. Electronically steerable phased array microphones can use the head position information to reject environmental noise. This provides the signal-to-noise gain necessary for remote microphones to be useful for speech recognition techniques [2]. Active cameras can also take advantage of up-to-date information about body part position to make fine distinctions about facial expression, identity, or hand posture.[6]

2.6 Audio Perception

Recently, we have been moving away from commercial recognizers and working with the details of the audio signal. In the Self-Awear system, a wearable computer clusters dynamic models of audio and video features to find consistent patterns in the data. This allows the system to differentiate between environments that appear similar yet sound different (and vice versa) [5]. In the TOCO project, a robotic bird interacts with a user to learn the names of objects and their properties. The system has no prior knowledge of English except for phonemes, and the user speaks to it in complete sentences (e.g., "this is a red ball"). The system then uses consistent cooccurrences in the two modalities to determine what acoustic chunks are associated with what visual properties. As a result, it is able to successfully extract nouns and adjectives corresponding to object names, shapes, and colors [9].

In current work, we are trying to make use of prosodic information to understand the cues of natural speech, both in the audio (pitch, energy, timing) and the visual (head motions, expressions) domains. Whereas speech recognition has focused almost entirely on the dictation task, in which speech is spoken evenly and follows the rules of grammar, we are interested in the situations involving natural interactions between users or between a user and a machine. Though grammar no longer applies to this situation, the audio visual cues (changes in pitch, whether the head is facing the agent, etc.) should provide the necessary information to direct the receiver's understanding of speech[3].

3 Perceptive Spaces

Unencumbered interface technologies do not, by themselves, constitute an interface. It is important to see how they come together with the context of an application. This section describes several systems that have been built in our lab. as well as ongoing work. They illustrate a progression from early audio-visual interfaces employing a low amount of interaction between the modalities to current work on more complex modal integration.

3.1 SURVIVE

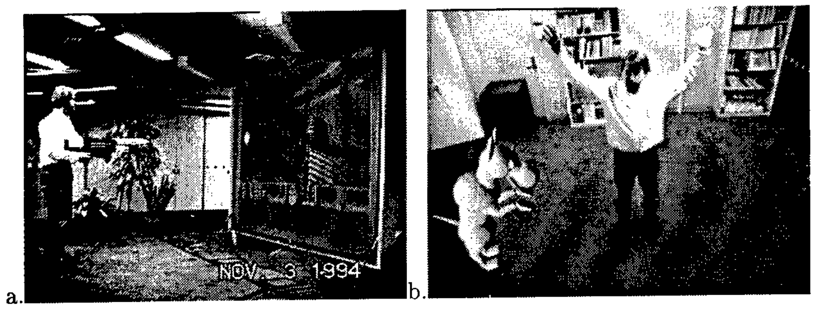


Fig. 6. (a) The user environment for SURVIVE. **(b)** User playing with the virtual dog in the ALIVE space

The first smart space application to employ both audio and visual perception was the entertainment application SURVIVE (Simulated Urban Recreational Violence Interactive Virtual Environment). Figure 6a shows a user in SURVIVE.

The user holds a large (two-handed) toy gun, and moves around the IVE stage. Position on the stage is fed into Doom's directional velocity controls. The hand position features are used to drive Doom's rotational velocity control. We built a simple IIR filterbank to discriminate between the two sounds produced by the toy gun. The results of the matched-filter provides control over weapon changes and firing. The vision system is used to constrain the context for the audio processing by only operating when a user was detected on the IVE stage.

Although simplistic, this interface has some very important features: low lag, intuitive control strategy, and a control abstraction well suited to the task. The interface processing requires little post-processing of the interface features, so it adds very little lag to the interface. Since many games have velocity-control interfaces, people adapt quickly to the control strategy because it meshes with their expectations about the game.

3.2 ALIVE

ALIVE combines autonomous agents with an interactive space. The user experiences the agents through a "magic-mirror" paradigm (including hamster-like creatures, a puppet, and a well-mannered dog—Figure 6b). The interactive space mirrors the real space on the other side of the projection display, and augments that reflected reality with the graphical representation of the agents and their world (including a water dish, partitions, and even a fire hydrant).

The "magic-mirror" model is attractive because it provides a set of domain constraints which are restrictive enough to allow simple vision routines to succeed, but is sufficiently unencumbered that is can be used by real people without training or a special apparatus.[8]

ALIVE employed a gesture-language that allowed the user to press buttons in the world or communicate wishes to the agents. ALIVE also employed audio perception. A commercial speech recognizer was used to turn speech events into commands for the agents. In this way speech provided a redundant modality to communicate with the agents.

Commercial speech recognizers require a very clean audio signal. It was critical to maintain the "hands-free" aspect of the interface, so we were unwilling to use a wireless headset or other such solution. Instead, we used a phased array of microphones that could be electronically steered to emphasize input from the user. The orientation of the steering was driven by the vision system, which could reliably track the user position [2]. In this way, the two modalities cooperated at the signal level.

3.3 City of News

City of News is an immersive, interactive web browser that makes use of people's strength remembering the surrounding 3D spatial layout. For instance, every-

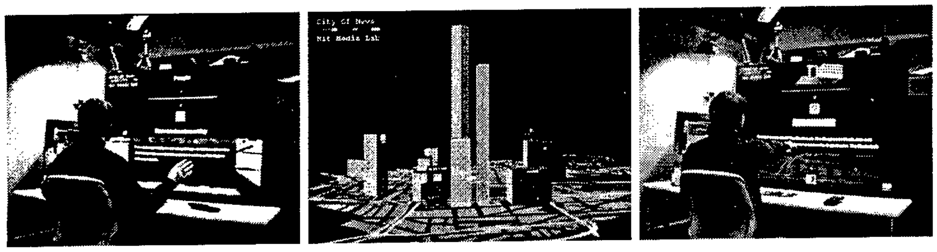


Fig. 7. City of News.

one can easily remember where most of the hundreds of objects in their house are located. We are also able to mentally reconstruct familiar places by use of landmarks, paths, and schematic overview mental maps. In comparison to our spatial memory, our ability to remember other sorts of information is greatly impoverished. City of News capitalizes on this ability by mapping the contents of URLs into a 3D graphical world projected on the large DESK screen. This virtual world is a dynamically growing urban landscape of information which anchors our perceptual flow of data to a cognitive map of a virtual place. Starting from a chosen "home page" - where home is finally associated with a physical space - our browser fetches and displays URLs so as to form skyscrapers and alleys of text and images through which the user can navigate. Following a link causes a new building to be raised in the district to which it belongs, conceptually, by the content it carries and content to be attached onto its "facade".

By mapping information to familiar places, which are virtually recreated, we stimulate association of content to geography. This spatial, urban-like, distribution of information facilitates navigation of large information databases, like the Internet, by providing the user with a cognitive spatial map of data distribution. This map is like an urban analogue to Yates' classical memory-palace information memorization technique.

To navigate this virtual 3D environment, users sit in front of the SMART DESK and use voice and hand gestures to explore or load new data. (Figure 7). Pointing to a link or saying "there" will load the new URL page. The user can scroll up and down a page by pointing up and down with either arm, or by saying "up/down". When a new building is raised and the corresponding content is loaded, the virtual camera of the 3D graphics world will automatically move to a new position in space that constitutes an ideal viewpoint for the current page. Side-pointing gestures, or saying "previous/next", allows to navigate along an information path back and forth. Both arms stretched to the side will show a full frontal view of a building and its contents. Both arms up drive the virtual camera up above the City and give an overall color-coded view of the urban information distribution. All the virtual camera movements are smooth interpolations between "camera anchors" that are invisibly dynamically loaded in the space as it grows. These anchors are like rail tracks which provide optimal viewpoints and constrained navigation so that the user is never lost in the virtual world.

The browser currently supports standard HTML with text, pictures and M-PEG movies. City of News was successfully shown at the Ars Electronica 97 Festival as an invited presentation.

A phased array was not necessary here since the user was seated at a desk, but the coupling of the gesture and speech modalities was critical to making a robust, usable interface. The visual cues are used to activate the commercial speech system, thus avoiding false recognitions during speech not addressed to the system. Speech in turn is used to disambiguate gestures[12].

3.4 The Perceptive Dance and Theater Stage

We have built an interactive stage for a single performer which allows us to coordinate and synchronize the performer's gestures, body movements, and speech, with projected images, graphics, expressive text, music, and sound. Our vision and auditory sensors endow digital media with perceptual intelligence, expressive and communicative abilities, similar to those of a human performer (Media Actors). Our work augments the expressive range of possibilities for performers and stretches the grammar of the traditional arts rather than suggesting ways and contexts to replace the embodied performer with a virtual one [11].

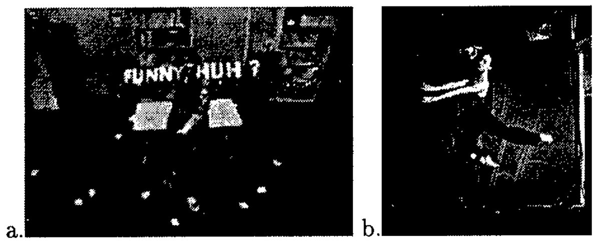


Fig. 8. (a) Improvisational performer Kristin Hall in the Perceptive Stage, at the MIT Media Lab, during the Digital Life Open House, on March 11, 1997. **(b)** Performer Jennifer DePalo, from Boston Conservatory, in DanceSpace, during rehersal.

In Improvisational TheaterSpace, Figure 8a, we create a situation in which the human actor can be seen interacting with his own thoughts in the form of animated expressive text projected on stage. The text is just like another actor able to understand and synchronize its performance to its human partner's gestures, postures, tone of voice, and words. Expressive text, as well as images, extend the expressive grammar of theater by allowing the director to show more of the character's inner conflicts, contrasting action/thought moments, memories, worries, desires, in a way analogous to cinema. A pitch tracker is used to "understand" emphasis in the perfromer's acting, and its effects are amplified or contrasted by the expressive text projected on stage. The computer vision's feature tracker is then used to align the projection with the performer. Gesture recognition gives start/stop/motion cues to the Media Actors.

Improvisational Theater Space followed research on DanceSpace, a computer vision driven stage in which music and graphics are generated on the fly by the dancer's movements, Figure 8b.

3.5 Netrek Collective

Netrek is a game of conquest with a Star Trek motif. Netrek is very much a team-oriented game. Winning requires a team that works together as a unit. This fact, in particular, provides a rich set of interface opportunities ranging from low-level tactics to high-level strategy. The Netrek Collective is an example of our current work toward interfaces with more cross-modal integration.

The first version of the Netrek Collective, entitled *Ogg That There*, is intended to perform in a manner similar to the classic interface demo "Put That There"[4]. Imperative commands with a subject-verb-object grammar can be issued to individual units. These commands override the robots internal action-selection algorithm, causing the specified action to execute immediately. Objects can either be named explicitly, or referred to with deictic gestures combined with spoken demonstrative pronouns. Figure 1 depicts a user selecting a game object with a deictic gesture.

Much like City of News, deictics are the only form of gesture supported by *Ogg That There*. They are labeled by speech events, not actually recognized. The grammar is currently implemented as a finite state machine, and speech recognition is accomplished with an adaptive speech recognizer developed in the lab[9].

Ogg That There succeeded in solving many integration issues involved in coupling research systems to existing game code. Current work involves redesigning the interface to more accurately match the flexibility of the perceptual technologies, the pace of play, and the need for a game interface to be fluid and fun.

This will mean even richer interaction between gesture and speech. The biggest challenges in this work are: the integration of the significant game context with speech and gesture to provide a more robust and expressive interface. This will involve combining the perceptual tools discussed in Sections 2.4 and 2.6 with a dynamic constraint system linking these perceptual signals to the changing game context.

4 Conclusion

Throughout these projects, we have always tried to take advantage of the coupling of speech with other modalities. Our impression is that it is only by exploiting the connections between such domains that we can hope to construct truly natural interfaces. Though the various pieces of our systems have become more complex over time, this philosophy continues to be an important factor in our continued work.

References

1. Ali Azarbayejani and Alex Pentland. Real-time self-calibrating stereo person tracking using 3-D shape estimation from blob features. In *Proceedings of 13th ICPR*, Vienna, Austria, August 1996. IEEE Computer Society Press.
2. Sumit Basu, Michael Casey, William Gardner, Ali Azarbayejani, and Alex Pentland. Vision-steered audio for interactive environments. In *Proceedings of IMAGE'COM 96*, Bordeaux, France, May 1996.
3. Sumit Basu and Alex Pentland. Headset-free voicing detection and pitch tracking in noisy environments. Technical Report 503, MIT Media Lab Vision and Modeling Group, June 1999.
4. R. A. Bolt. 'put-that-there': Voice and gesture at the graphics interface. In *Computer Graphics Proceedings, SIGGRAPH 1980,*, volume 14, pages 262–70, July 1980.
5. Brian Clarkson and Alex Pentland. Unsupervised clustering of ambulatory audio and video. In *ICASSP'99*, 1999.
6. T. Darrell, B. Moghaddam, and A. Pentland. Active face tracking and pose estimation in an interactive room. In *CVPR96*. IEEE Computer Society, 1996.
7. M. W. Krueger. *Artificial Reality II.* Addison Wesley, 1990.
8. Pattie Maes, Bruce Blumberg, Trevor Darrell, and Alex Pentland. The alive system: Full-body interaction with animated autonomous agents. *ACM Multimedia Systems*, 5:105–112, 1997.
9. Deb Roy and Alex Pentland. "learning words from audio-visual input. In *Int. Conf. Spoken Language Processing*, volume 4, page 1279, Sydney, Australia, December 1998.
10. Kenneth Russell, Thad Starner, and Alex Pentland. Unencumbered virtual environments. In *IJCAI-95 Workshop on Entertainment and AI/Alife*, 1995.
11. Flavia Sparacino, Christopher Wren, Glorianna Davenport, and Alex Pentland. Augmented performance in dance and theater. In *International Dance and Technology 99*, ASU, Tempe, Arizona, February 1999.
12. C. Wren, F. Sparacino, A. Azarbayejani, T. Darrell, James W. Davis, T. Starner, Kotani A, C. Chao, M. Hlavac, K. Russell, Aaron Bobick, and Pentland A. Perceptive spaces for performance and entertainment (revised). In *ATR Workshop on Virtual Communication Environments*, April 1998.
13. Christopher Wren, Ali Azarbayejani, Trevor Darrell, and Alex Pentland. Pfinder: Real-time tracking of the human body. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19(7):780–785, July 1997.
14. Christopher R. Wren and Alex P. Pentland. Dynamic models of human motion. In *Proceedings of FG'98*, Nara, Japan, April 1998. IEEE.

A Tourist-Centric Mechanism for Interacting with the Environment

M.J. O'Grady, R.P. O'Rafferty, G.M.P. O'Hare

Practice & Research in Intelligent Systems & Media (PRISM),
Dept. of Computer Science, University College, Dublin (UCD), Belfield, Dublin 4, Ireland.
{ Michael.J.OGrady, Ronan.ORafferty, Gregory.OHare } @ucd.ie

Abstract. Facilitating interaction between users and their environment has been the subject of much research in computer science. One practical application that evolves from this research is that of an intelligent tourist assistant. In this paper, HIPS, an innovative implementation of such an application is briefly outlined. In particular, we describe an implementation that exploits HIPS concepts to enable intelligent interaction between tourists and their environment.

1 Introduction

Within this paper we present a tourist-centric mechanism for environment interaction. This application is hosted on a handheld computing device in the form of a Personal Digital Assistant (PDA) which presents a contextualised information delivery experience based upon location aware and user aware information.

The environment in question may be either interior or exterior, in the context of this paper we concentrate on the latter, typically of the scale of a cultural or tourist quarter of a city. Location aware data is captured through the use of a Global Positioning System (GPS) while user aware data is available through a user profile.

We seek to give the illusion of an intelligent environment sensing the information needs of the user. In reality it is inconceivable to truely pepper such an external environment with the necessary sensing infrastructure. However it is possible to deliver adaptivity, personalisation and contextualisation based on the location and user aware mechanisms hosted upon the PDA.

HIPS (Hyper Interaction within Physical Space) is an application being developed by the HIPS consortium for the intelligent dissemination of information to tourists [1]. Our work is being carried out under the auspices of the HIPS initiative.

Section 2 situates this work in a broader research landscape. Section 3 provides an overview of the HIPS architecture while Section 4 describes in particular work conducted at UCD on the implementation of a mechanism for facilitating interaction between tourists and their environment. Finally Section 5 presents a discussion and the plans for future work.

2 Related Work

Much of the research carried out in smart environments has its origins in AI and utilises techniques such as video-based perception and speech understanding. Pentland *et al.* [2] and more recently Bobick *et al.* [3] have investigated smart environments. This research has led to the development of environments that use video based perception to achieve object tracking, movement detection and action recognition of people in the environment. Using such remote sensing techniques enables unencumbered activity as users are not required to carry or wear any particular equipment. Coen [4][5] and Torrance [6] also use similar AI techniques in their work on Intelligent rooms.

Abowd and Brotherton [7] have also researched the area of Smart Environments in the ClassRoom 2000 project. This classroom takes media-enhanced notes on behalf of the students which allows them to concentrate on the lecture rather than just record the lecture.

The research of Foster *et al.* [8][9] on Intelligent Buildings attempts to provide navigation and information services around intelligent buildings. They use an existing sensor control network and smart card transducers to provide information, visual and aural, to aid navigation of large buildings.

The areas of smart environments and ubiquitous computing overlap in their need to sense events in the environment, research into the use of sensor networks to determine user position was one of the early research interests of ubiquitous computing.

Weiser [11] defined ubiquitous computing as follows: "computer usage is enhanced by making many computers available throughout the physical environment, while making them effectively invisible to the user". The Active Badge Location System by Want *et al.* [10] was developed as a method of locating staff within the office environment. This would allow for example, easy routing of phone calls to the phone nearest the active badge wearer. Another ubiquitous computing application is the ParcTab, it is similar to the active badge in that it can locate the owner, but also the ParcTab has a pen-based interface with which the user can access remote applications. In both the active badge and the ParcTab the user wears a device which emits an infrared (IR) signal, the users position is determined by a networked array of IR transceivers that sense the users IR signal.

Schilit *et al.* [12] further developed the use of ParcTabs and its positioning capability to investigate context-aware computing applications. According to Schilit, context aware computing "can promote and mediate people's interactions with devices, computers and other people and also help navigate unfamiliar places". To support further research in context and location aware applications Abowd *et al.* have developed the Context Toolkit [13] which provides re-useable components, thus facilitating quick development of context aware applications.

As can be seen, many of these systems are suited to physically small environments (i.e. room, office, building). However, large environments (i.e. campus and city size

environments) require alternative techniques for environmental interaction, as it is impractical to fully duplicate indoor infrastructure in an outdoor scenario. For an outdoor environment it is necessary that the user be equipped with some context-aware computing device to exploit system services.
To date research into this area has concentrated on the development of wearable computers [14][15] and of location aware tourist information systems [18]. Long *et al.* [16] developed Cyberguide a mobile context aware tour guide for both indoor and outdoor environments that delivers maps and other location sensitive information. Davies *et al.* [17] have developed GUIDE, a context sensitive tourist guide for visitors to the city of Lancaster.
Currently we are involved in the HIPS project, which attempts to develop a handheld tour-guide that delivers multi-media presentations that are dynamically adapted [19] both to the users context and to an individuals user model [20][21]. In contrast with other work HIPS utilises content adaptation and user modelling techniques.
Our objective within HIPS is to utilise the HIPS infrastructure in an outdoor environment, ensuring that the functionality of HIPS can be easily duplicated in new exterior environments.

3 The HIPS System

HIPS is an innovative system for delivering context sensitive information to users. Visitors to a location, for example a museum or an archeological site, equipped with HIPS enabled devices will be supplied with personalised multi-media information specific to their position and relevant to their individual needs.
A fundamental objective of HIPS is to provide an infrastructure which supports simultaneous user navigation within both a physical space and its corresponding information space, with particular emphasis on the needs of a tourist. HIPS is aware of individual user interest and experience, and adapts its presentations accordingly. In this way, a HIPS presentation should be a more enlightening experience for the user.
The core elements of HIPS are, therefore, location awareness and information personalisation. Naturally, there is considerable focus on ease of use and implementation transparency.

3.1 HIPS Components

The HIPS architecture is based on a client-server model. Both of these components are now described.

HIPS Client
A HIPS client, known affectionately as a hippy, is a handheld computer which supports pen-based input and is capable of displaying multimedia presentations. In addition to this, it must facilitate wireless connections to the HIPS server, either via GSM or a local wireless LAN, and contain a localisation or position sensing mechanism supported by HIPS. At present, IR is used for interior position

determination and GPS for exterior. This localisation component follows a plug and play ethos so that moving from an indoor to an outdoor environment requires little or no manual configuration.

HIPS clients supports two modes of interaction:

- Implicit: As the user moves around the physical environment, the sensor data changes and this triggers an interaction with the server which determines if a new presentation is required.
- Explicit: The user has the option of requesting more information on a particular exhibit. Alternatively, a presentation may be cut short if the user so desires.

While it is intended to develop an actual physical device for utilising HIPS, any device that contains the appropriate hardware and software should be able to access the HIPS system.

HIPS Server

The HIPS server is a sophisticated system for supporting roaming HIPS clients. It contains a number of complex components including:

- User Model – The user model contains knowledge pertaining to information that has already been presented to the user, user interests, user interaction history and user movement in the physical environment.
- Database – Models of the physical world and their corresponding information spaces are stored in the database. A repository of multimedia objects used for user presentations is also maintained by the database.
- Presentation Planner – By consulting the user model and the database, contextualised and personalised presentations are dynamically assembled and dispatched to the client.

Scenarios for Utilising HIPS

Indoor Scenario

An indoor prototype has been developed by the HIPS consortium for the Museo Civico in Siena, Italy. As the user explores the museum, commentaries about encountered artifacts are dynamically generated. Depending on user interests and expertise, the system can recommend other exhibits that may be of particular interest.

Outdoor Scenario

Having successfully demonstrated that the core HIPS technology works in an indoor environment, key components have been consumed and extended so that the system functions outdoors.
The reminder of this paper describes the exterior prototype and its supporting infrastructure in more detail.

4 The Location Aware Tour Guide

4.1 Introduction

The success of HIPS necessitates the dissemination of the results and the potential of the work in a wide variety of arenas. This specifically demands that:

- The localisation technology we use should not have a high infrastructure setup cost, therefore we wish to make maximum use of existing localisation infrastructure (e.g. GPS).
- We need to be able to rapidly develop local tour-guides for new tourist information domains (e.g. different cities across Europe). Therefore the system must equip an administrator with a powerful set of tools to populate the database with the location dependent information.

We are consuming the components from the indoor prototype that realise presentation personalisation (i.e. presentation assembler and user model) and also introduce GPS as the localisation technology.

4.2 System Components

The architecture of the system can be logically divided into four layers: location layer, presentation layer, server layer and data layer. Each of these layers and their constituent components can be seen in figure 1 and are now examined in detail.

Location Layer

The primary objective of this layer is to combine localisation sensors with a software module to interpret the received data and to communicate this information to the appropriate control components. Position is determined using a GPS receiver, namely the Garmin GPS II+, the output of which complies with the NMEA 0183 v2.0 standard [22]. Position readings from the receiver are sent every 2/3 seconds via a serial cable to a localisation daemon running on the client. This daemon is a signed Java applet which maintains a connection to the serial port of the client through the Java serial communications API. When new information becomes available, the NMEA compliant data is parsed to retrieve the global position sentences, this data is then dispatched to the server.

GPS as a localisation technology with an accuracy of less that 50 meters has proved inadequate for our requirements. Coupled with the poor accuracy, GPS is affected by signal degradation in environments with a high density of buildings (e.g. a city center). To overcome these difficulties we are investigating a number of solutions to supplement and complement GPS to achieve a more accurate positional lock. These include:

- Differential GPS, which provides a sub-10 meter accuracy but can be influenced by the same factors that adversely affect GPS.
- Utilising existing cellular infrastructure to refine position, (i.e. each base station in a cellular network has a known geographical position which may be used to enhance position determination).
- Use of an electronic compass to supplement location information with orientation for more accurate selection of relevant information.

The fusion of such diverse technologies will allow position to be determined more accurately, and consequently increase the effectiveness of the system.

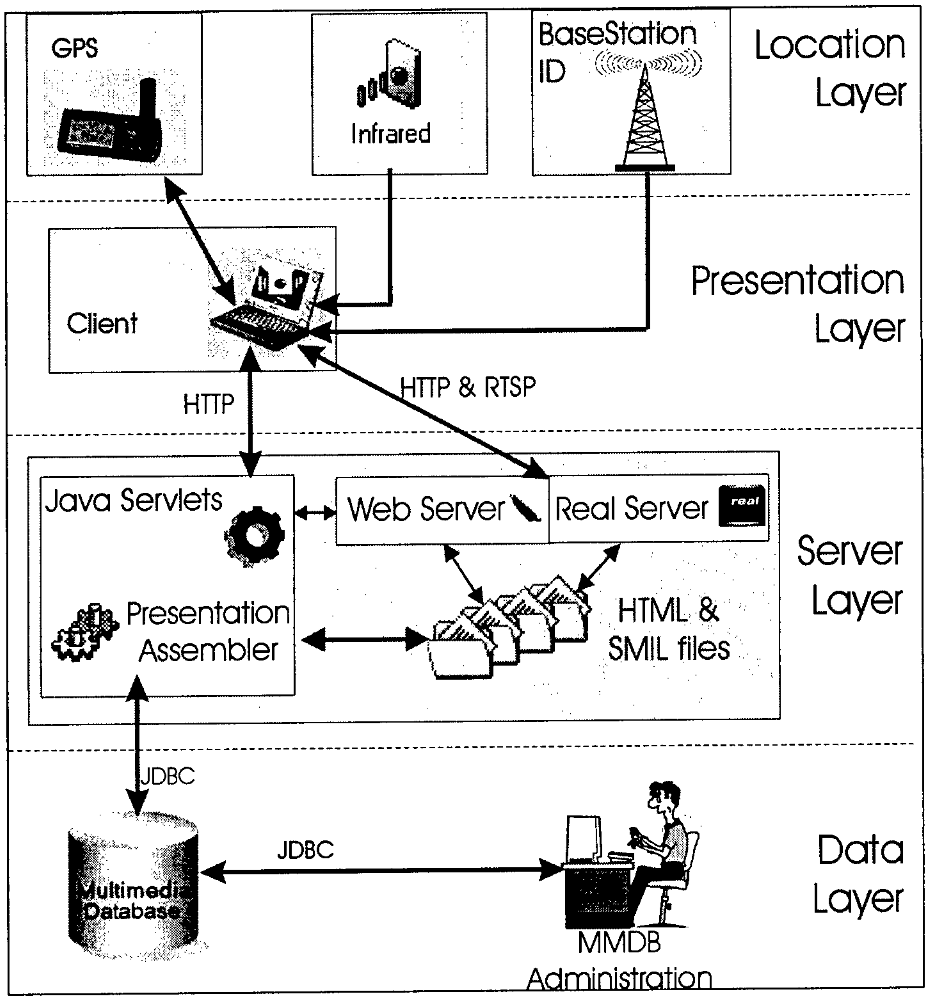


Fig. 1. The Outdoor HIPS Architecture

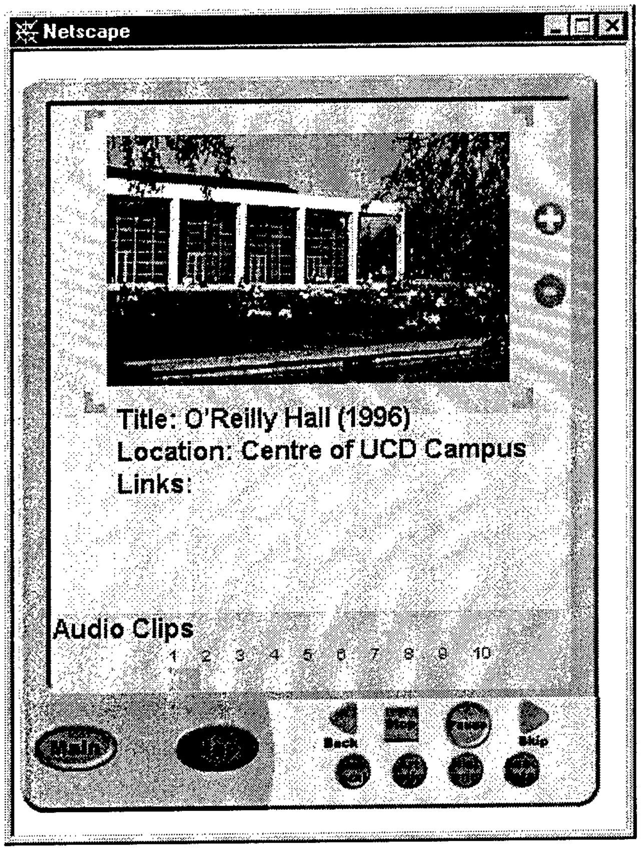


Fig. 2. The PDA User Interface

Presentation Layer

The Presentation layer contains the user interface to the system and runs on a Toshiba Libretto 110. The interface, which can be seen in figure 2, was designed and developed by CB&J [23] for HIPS and is similar to the user interface used for the indoor HIPS tour-guide. It is not coincidental that the interface dimensions correspond to the screen size of an average PDA. When palm-sized PCs become more powerful this interface should be easily ported to a PDA platform.

The user interface was developed with JavaScript and HTML and runs in a standard web browser. The browser combined with a RealPlayer plug-in provide the medium through which multi-media presentations can be delivered. The multi-media presentations are composed of HTML, JPEG, SMIL (Synchronised Multimedia Integration Language) and RealAudio files.

Server Layer
The server layer consists of a web server, RealAudio server and a set of Java Servlets. We use the Apache web server with the Apache JServ extension to enable execution of Java servlets. The servlets manage the connection with the client and also assemble new presentations.
The presentation assembler servlet manages the preparation of presentations and the notification of the client. When a new presentation is required, the multi-media database is queried and the appropriate multi-media objects are extracted. Using these objects an HTML file, and a SMIL [24] file are dynamically created. SMIL is a W3 Consortium standard that facilitates synchronised multimedia presentations on the Web while lowering the bandwidth requirements for transmitting this type of content. This enables us to dynamically assemble audio commentaries of varying length and content. Once the files have been generated and written to the appropriate directories the Presentation layer is notified that a new presentation is available.

Two other important tasks include:

- Media anticipation: Attempt to predict which media object should be preloaded to the client, this could be used to mask latencies of a low bandwidth communication system (e.g. GSM).
- Visit memory: Knowledge of previously presented information is retained to prevent duplicate presentation delivery.

Data Layer
To support an information infrastructure of this nature, an industrial strength database is required. The capability to support multimedia data types is essential. For this prototype, we used IBM DB2 Universal Database. The ability to easily integrate DB2 and Java using JDBC was an added advantage.

Database Design
All data maintained in the multimedia database may be classified under two headings:

- Position-related
  A number of tables in the database define the relationship between a physical location (Latitude & Longitude) and a logical location (e.g. Leaning tower of Pisa). This relationship is fundamental to ensuring successful contextualised interaction between a tourist and the corresponding information space (see figure 3). As it is more intuitive to use logical location rather than the physical location, logical location will therefore be a primary key for other tables in the database.

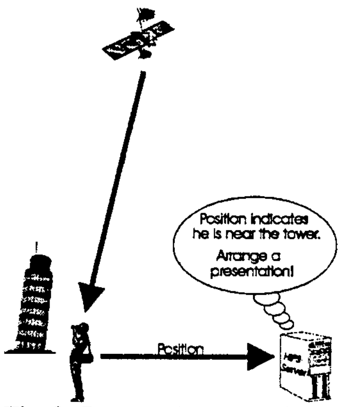


Fig. 3. Determining Logical Location

- Media–related
  Any presentation can consist of a combination of audio, video, image and text objects. When a presentation is being assembled (see figure 4), the appropriate objects are identified using the logical position as key and then extracted from the database. The presentation is then dispatched to the user.

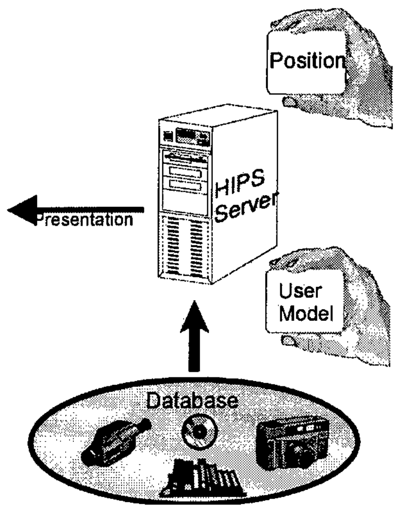


Fig. 4. Determining Presentation

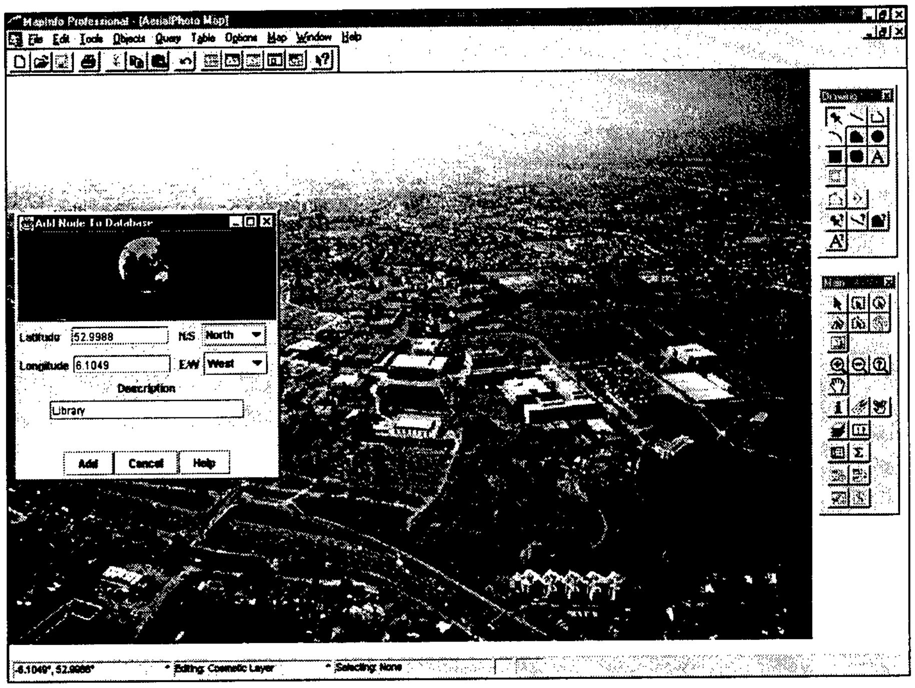
The user model and other components use their own propriety data formats, however, it is planned to ultimately use one common database for all components.

There are a number of advantages obtained by separating the positional data from the media data. In so doing:

- It allows for future enhancements to the localisation mechanism. For example, if an innovative wireless technology were to be developed, it would be a simple case of relating the wireless beacons to the corresponding physical objects. The remaining data tables and associated access routines need not be modified. In addition to this, it is likely that more sophisticated position detection algorithms may be developed, for example the use of orientation as an additional parameter would enhance the system.
- It facilitates the distribution of the database as it may not always be desirable to have all the data in one physical repository. For example, it may be appropriate in some situations to have all calculations regarding position determination performed on the client rather than on the server. Also, if the system were integrated with a modern mobile network, the database could be distributed amongst the constituent Base Stations.

Database Administration
The development of a comprehensive suite of database administration tools is a key requirement for facilitating the rapid deployment of new information spaces. Two methods of administrating the database have been designed.

- GIS-Based Local Administration:
  A Geographic Information System (GIS) is the logical tool to use when manipulating geographic data. Strenuous efforts have been made recently by most GIS vendors to integrate their products with popular commercial database systems and the internet. Also, there is an initiative [25] to make GIS data more accessible to third party software.
  The key prerequisite for using a GIS is the availability of a digital map. Using this, the physical position of some object can be read directly from the map and immediately registered in the database. The corresponding multimedia objects are then associated with this position and the database is updated accordingly. A snapshot of this database administration tool is shown in Figure 5.


MapInfo Professional - [AerialPhoto Map]
File Edit Tools Objects Query Table Options Map Window Help
Add Node To Database
Add
Cancel
Help

Latitude 52.9988 N/S North
Longitude 6.1049 E/W West
Description
Library

-6.1049°, 52.9988° Editing: Cosmetic Layer Selecting: None

Fig. 5. Using a GIS for position determination

- Web-Based Remote Administration:
  A simple web-based interface has been designed for adding material to the database. It is anticipated that, using this mechanism, administrators can physically go to some new area for which they want to develop an information space and build it in situe. All that is required is a GPS unit for geographical position identification, a digital camera for photographs and a laptop with both sound recording facilities and internet connection capabilities. Once the information space has been defined, and all the media components assembled, the central database can be updated. The interface is similar but simpler than that used by the GIS.

5 Conclusion and Future Work

The work described here presents a mechanism through which the tourist can interact with the exterior environment. To date two demonstrators have been developed, one based on a UCD campus tour guide and the other on a tour guide for Piazzo Del Campo in Siena.
Initial tests indicate that GPS, in certain external environments may prove inadequate due to signal degradation. Therefore we intend augmenting our current localisation mechanism with additional technologies, such as differential GPS, electronic compasses, or utilising existing cellular infrastructure to refine position. In order to test the system under true thin client conditions we are currently in the process of integrating the system with GSM.
Within the HIPS initiative we intend to adapt the infrastructure so that both indoor and outdoor localisation technologies can co-exist and function in a manner that is transparent to the user. We anticipate development of a technique through which tourists can annotate the physical space (i.e. visitors to a location can record their observations and these can be accessed by future visitors). This is akin to a *collective visit memory* achieved through a form of *cyber graffiti.*
In the long term we plan to integrate the system with an actual mobile network, using advanced technologies such as the Wireless Application Protocol (WAP).
Currently we are investigating system scalability issues in migrating from a campus prototype to a city wide application.

6 Acknowledgements

We would like to thank and acknowledge all members of the HIPS consortium for their active support and encouragement. The HIPS consortium includes: University of Siena (Italy), University of Edinburgh (UK), CB&J(France), GMD(Germany), IRST(Italy), SIETTE-Alcatel(Italy), SINTEF(Norway) and University College, Dublin (Ireland). We gratefully acknowledge the support of Esprit LTR through project No. 25574 under the I^3 net (Intelligent Information Interfaces) initiative.

7 References

1. HIPS (Hyper Interaction within Physical Space) WWW home page for the HIPS project : http://www.ing.unisi.it/lab_tel/hips/hips.html
2. Pentland, A.: Smart Rooms. Scientific American, 274(4):68-76, April 1996.
3. Bobick, A., Intille.,S., Davis, J., Baird, F., Pinhanez, C., Campbell, L., Ivanov, Y., Schutte, A., Wilson, A.: The KidsRoom: A Perceptually-Based Interactive and Immersive Story Environment. Appears in: Presence: Teleoperators and Virtual Environments, 8(4), August 1999. pp. 367-391.
4. Coen, M.: Design Pricipies for Intelligent Environments. In Proceedings of The Fifteenth National Conference on Artificial Intelligence. (AAAI 98) Madison, Wisconsin 1998.

5. Coen, M.: The Future of Human-Computer Interaction or How I learned to stop worrying and love my Intelligent Room. IEEE Intelligent Systems March/April 1999.
6. Torrance, M. C.: Advances in Human-Computer Interaction: The Intelligent Room. In Working Notes of the CHI 95 Research Symposium, May 6-7, 1995, Denver, Colorado.
7. Brotherton, J. A., Abowd, G. D.: Rooms Take Note: Rooms Takes Notes!. In Working Papers of AAAI '98 Spring Symposium, March, 1998.
8. Foster, G.T.: The ARIADNE Project: Supporting Access, Navigation and Information Services in Labyrinth of Large Building. Proceedings of the 4th European Conference for the Advancement of Assistive Technology AAATE'97. Thessaloniki, Sept 1997.
9. Foster, G.T., Glover J.P.N.: Supporting Navigation Services Within Intelligent Buildings. Proceedings of the First Mobinet Symposium. Athens, May 1997.
10. Want, R., Hopper, A., Falcao, V., Gibbons, J.: The Active Badge Location System. ACM Transactions on Information Systems, Vol. 10, No. 1, pages 91-102, January 1992.
11. Weiser, M.: Some computer science issues in ubiquitous computing. Communications of the ACM, 36(7):75-84, July 1993.
12. Schilit, B., Adams, N., Want, R.: Context-Aware Computing Applications. Proceedings of Workshop on Mobile Computing Systems and Applications. Pages 85-91. 1994.
13. Salber, D., Dey, A. K., Abowd, G.A.: The Context Toolkit: Aiding the Development of Context-Enabled Applications. In Proceedings of CHI'99, Pittsburgh, PA, May 15-20, 1999.
14. Starner, T., Mann, S., Rhodes, B., Levine, J., Healey, J., Kirsch, D., Picard, R., Pentland, A.: Augmented Reality Through Wearable Computing. Appears in Presence 6(4), 1997.
15. Feiner, S., MacIntyre, B., Höllerer, T., Webster, A.: A Touring Machine: Prototyping 3D Mobile Augmented Reality Systems for Exploring the Urban Environment. In Proceedings of the 1st International Symposium on Wearable Computers (ISWC '97), Cambridge, Massachusetts, October 13-14, 1997.
16. Long, S., Kooper, R., Abowd, G.D., Atkeson, C. G.: Prototyping of Mobile Context-Aware Applications: The Cyberguide Case Study. Proceedings of the 2nd ACM International Conference on Mobile Computing (MOBICOM), Rye, New York, U.S., Pages 97-108. 1996.
17. Davies, N, K. Cheverst, K. Mitchell and A. Friday.: 'Caches in the Air': Disseminating Tourist Information in the Guide System. In Proceedings of 2nd IEEE Workshop on Mobile Computer Systems and Applications (WMCSA '99). New Orleans, Louisiana, 25 - 26 February 1999.
18. O'Rafferty, R.P., O'Grady, M.J., O'Hare, G.M.P.: A Rapidly Configurable Location-Aware Information System for an Exterior Environment. To appear International Symposium on Handheld and Ubiquitous Computing (HUC 99), Karlsruhe, Germany, September 27-29, 1999.
19. Petrelli, D., Not E., Zancanaro M.: Getting Engaged and Getting Tired: What Is in a Museum Experience. In the proccedings of the Workshop on 'Attitude, Personality and Emotions in User-Adapted Interaction' held in conjunction with UM'99, Banff, 23 June 1999.
20. Oppermann, R., Specht, M.: Adaptive support for a mobile museum guide. In the Proceedings of Interactive Applications of Mobile Computing, IMC'98, Rostock, Germany - November 24-25, 1998.
21. Marti, P., Rizzo, A., Petroni, L., Tozzi, G., Diligenti, M.: Adapting the museum: a non-intrusive user modeling approach. Proceedings of the Seventh International Conference on User Modeling, UM99. pages 311-313. Banff, Canada, June 20-24, 1999.
22. NMEA. NMEA 0183 Interface Standard, March 1998.
23. Broadbent, J.: HIPS User Interface Specification. HIPS Internal Report, 1999
24. W3 Consortium homepage : http://www.w3.org/ .
25. Open GIS homepage : http://www.opengis.org/ .

A Context Sensitive Natural Language Modality for an Intelligent Room

Michael Coen, Luke Weisman, Kavita Thomas, and Marion Groh

MIT Artificial Intelligence Lab
545 Technology Square
Cambridge, MA 02139
{mhcoen, luke, marion}@ai.mit.edu, kavita@cs.cmu.edu

Abstract. This paper describes the design and implementation of a natural language interface to a highly interactive space known as the Intelligent Room. We introduce a data structure called a *recognition forest*, which simplifies incorporation of non-linguistic contextual information about the human-level events going on in the Intelligent Room into its speech understanding system. This aim of using context has been to allow multiple applications—all of which support a relatively high degree of natural syntactic variability—to run simultaneously in the Intelligent Room.

1 Introduction

This paper describes the design and implementation of the natural language interface to the MIT Artificial Intelligence Lab's Intelligent Room Project [3,5]. The Intelligent Room explores natural interaction with embedded computational systems. It has a host of computer vision and speech understanding systems that connect it to ordinary, human-level events occurring within it.

In this paper, we are concerned with the overall design and implementation of the Intelligent Room's support for natural language interactions. The main feature of this is the *recognition forest*, a linguistic data structure that simplifies incorporation of contextual information into the room's speech understanding system and allows the room's multiple applications to independently access its speech modality. Our motivation for using context is: to help manage the combinatorial explosion in processing time that followed the incorporation of natural syntactic variability into our speech recognition system; to allow for the incorporation of non-linguistic information into linguistic contextual model; to disambiguate diectic references; and to provide spoken language input to coexisting, independent applications.

In this paper, we present the recognition forest as a useful tool for creating spoken language interfaces to intelligent, interactive spaces. We will also discuss how it

This material is based upon work supported by the Advanced Research Projects Agency of the Department of Defense under contract number F30602—94—C—0204, monitored through Rome Laboratory.

contributed towards satisfying the goals that shaped the design of the Intelligent Room's natural language interface, including:

1. To support unimodal speech interactions, i.e. interactions that do not tie the user to a keyboard, mouse or display.
2. To leverage off very strong notions of context inherent in room applications to allow for better speech understanding.
3. To allow multiple speech-enabled room applications to coexist without a heavyweight central controller.
4. To provide for dynamic sets of recognized utterances.
5. To employ only very shallow linguistic knowledge and representations.

The decision to minimize the amount of linguistic knowledge contained in our system was made to facilitate the room's infrastructure and application development by a wide range of people, particularly computer science undergraduates who have no formal exposure to computational linguistics. Our system needed to be accessible by all researchers in the project, regardless of their background. We believer that many of the issues discussed in this paper will remain useful when applied to systems with more sophisticated linguistic representations. Given the increasingly widespread interest in highly interactive, computational environments [7], many other designers and implementers will be faced will similar challenges, and we hope our approach will be generally useful for other systems.

Over the past three decades there have been significant research efforts devoted to the development of natural language interfaces. We divide these into three distinct classes based on the modality chosen for natural language interaction. The first consists of text-only dialog systems, such as SHRDLU [12], Lunar [13], and the multitude of database query systems, such as START [8]. In these systems typewritten text is used for input and output. With the advent of improved speech recognition and synthesis, efforts were made to integrate these technologies into natural language dialog systems—our second category—such as those in [8,11,14]. However, with these the user is still expected to interact with the system at a terminal where the display of queries and recognition results are perused and then potentially disambiguated by the user. Many of these systems also make use of "push-to-speak button," in order to allow the user to signal the speech recognizer that the next utterance is to be treated as input to the system. The final class of systems is those that operate only in the speech modality, such as SUNDIAL [1], and Jupiter [3], two telephone communication systems for answering domain specific queries.

Our approach represents a significant departure from those described above. The choice of the ability to operate in a voice-only modality separates it from the text-only and mixed voice and screen systems. This departure requires an emphasis on careful planning and structuring of dialog to take advantage of speech while also overcoming some of the difficulties inherent in the speech modality. Our approach is different from that of the other voice-only language projects in that: we are not tackling applications that monopolize the user's attention or require excessive word knowledge; we allow people interacting with the Intelligent Room to readily change application contexts; we place more emphasis on speech recognition in noisy, multi-user environments where people are primarily talking to one another; we seek to plan interactions with minimum intrusiveness; and we provide an interface to multiple applications simultaneously. Our system is not intended to be task or domain specific; it is used in the same way as a keyboard and mouse—a general input device simultaneously used by many applications. It is a tool, not an end in itself.

In the next section we briefly motivate and describe the Intelligent Room as the platform and motivation for our work. Next, we present a user's perspective of the Intelligent Room. In section 4 we outline the room's computational architecture and linguistic systems. Then we give several representative linguistic interactions illustrating the concepts we have described. Finally, we discuss limitations inherent in our approach and possible remedies.

2 The Intelligent Room

We now proceed to briefly describe the Intelligent Room as the platform for our research. More in depth discussion of the room can be found in [6] and details of its multimodal resolution are contained in [5].

The Intelligent Room is a research platform for exploring the design of intelligent environments [7]. The Intelligent Room was created to experiment with different forms of natural, multimodal human-computer interaction (HCI) during what is traditionally considered non-computational activity. It is equipped with numerous computer vision, speech and gesture recognition systems that connect it to what its inhabitants are doing and saying. The motivation for researching intelligent environments is to bring computation into the real, physical world. The goal is to allow computers to participate in human-level activities that have never previously involved computation and to allow people to interact with computational systems the way they would with other people: via gesture, voice, movement, and context.

The Intelligent Room is a space populated by computer controlled devices; these include overhead LCD projectors and displays, audio/visual multiplexers, VCRs, drapes, blinds, stereos, steerable video cameras, etc. The video cameras are used by the room's computer vision systems. These vision systems are detailed in [6], but of relevance here are the following: a person tracking system that can locate people in real-time as they move about the room; gesture recognition of both finger and laser pointing on either of the room's projected LCD displays; and a system that specifically notices when people sit down on particular pieces of furniture.

Other research in intelligent environments [2,9,10] has focused more on development of computer vision and other sensing technologies at the expense of linguistic interactions. We believe, however, that language is fundamental to having meaningful and complex interactions with these sophisticated, interactive spaces. In particular, we are interested in speech understanding systems that function more like a language modality than a voice simulation of a keyboard or mouse. Although the Intelligent Room's current linguistic systems require a great deal of development before they near this goal, we believe our approach is an extensible first approximation.

3 User Interactions

People in the Intelligent Room wear wireless lapel microphones that transmit to the speech understanding system described below. By default, the room ignores the spoken utterances of its inhabitants, which are generally directed to other people within the room. This state is known as "the room being asleep."[1] To obtain the room's attention, a user stops speaking for a moment and then says the word "Computer." The room immediately responds with an audible, quiet chirp from an overhead speaker to indicate it is paying attention. The user then has a two second window in which to begin speaking to the room. If the room is unable to recognize any utterances starting within that period, it silently goes back to sleep until explicitly addressed again. However, if what the user says is recognized, the room responds with an audible click and then under most circumstances it returns to sleep. This hands- and eyes-free style of interaction coupled with audio feedback allows a user to ignore the room's computational presence until she explicitly needs to communicate with it. There is no need to do anything other than preface spoken utterances with the cue *Computer* to enable verbal interaction. Thus, a user can interact with the room easily, regardless of her proximity to a keyboard or monitor. Additionally, explicitly cueing the room minimizes the likelihood that extraneous speech or noise will incorrectly trigger a recognition event. Importantly, it also allows detection of non-recognition events, i.e. times when the room is not able to understanding something the user is explicitly trying to convey. In the event the room erroneously wakes up due to an incorrect recognition event, it will either go back to sleep automatically when the two second window expires or the user can explicitly tell it to "go to sleep" upon hearing the wake-up chirp.

When the room is awake, it is listening for a specific set of utterances contained in its recognition forest, a data structure described in the next section. Under appropriate circumstances, users can also freely and continuously dictate expressions to the room unconstrained by any grammar or rule set. This capability is used, for example, during information retrieval queries (such as a web search) for which it is unreasonable to expect that the room's grammars already contain the sought after phrase. In these interactions, the room repeats the final utterance back to the user to verify correct recognition. The lag time between user speech and room verification is extremely small, and this mode of interaction has proved to be quite useful provided input is short in length. We note here that the room is responsible for switching between the constrained and diction speech modes; users do not explicitly change this state.

The room can also remain awake listening for utterances. Someone intending on a prolonged series of verbal interactions can simply tell the room to "stay awake." The room continues to provide an audible click after recognized statements, but these statements no longer need to be preceded by the spoken *Computer* cue. In addition, the room can wake itself up if it expects an utterance from the user for some reason. For example, when the room asks the user a question, it will stay awake for several

[1] The room's vision systems continue to respond to users even when it is not listening for their verbal input.

seconds waiting for an answer. If that period ends without a response being given, the room provides an audio timeout signal to indicate that it is going back to sleep.

4 Speech Understanding

In this section we present the Intelligent Room's speech understanding system. We begin with a discussion of the interaction between the software agents that control the operation of the room and the room's set of recognition grammars.

The Intelligent Room is controlled by a modular system of approximately 100 distinct, intercommunicating software agents that run on several networked workstations. These agents' primary task is to connect various components of the room (e.g., vision and speech recognition systems) to each other as well as to internal and external stores of information (e.g., a person locator or an information retrieval system). Essentially, these agents are intelligent *computational glue* for interconnecting all of the room's components and moving information among them.

The Intelligent Room listens for continuous speech utterances contained in a forest (or set) of multiple grammars, which we call the recognition forest. Each grammar in the recognition forest is created by one of the room's software agents (see Figure 1), which receives notification when any utterance contained in one of its grammars is heard. Agents do not necessarily know nor need to know of any other grammars or agents. (We note for clarification that a single agent is allowed and actually encouraged to create multiple grammars.) The notification message for a recognized utterance contains a parse tree, which the agent can manipulate to determine its content. In the recognition forest, a grammar is called "active" if the room is currently listening for it and "inactive" otherwise. Active grammars are rank ordered in terms of their expected likelihood of being heard.

Fundamental to our design is that all of the grammars must be constrained to highly specific contexts among which some component of the Intelligent Room is capable of distinguishing. Instead of keeping a single enormous recognition grammar active, the room collectively keeps subsets of small grammars active in parallel, given what it currently expects to hear. The key assumption here is that certain types of

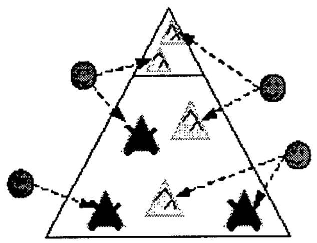


Fig. 1. Software agents creating the forest of context free recognition grammars. Each small triangle represents a grammar, and each face represents a software agent. Active grammars are lightly colored. Inactive grammars are crossed out. The uppermost, demarcated region contains universal grammars that are always active.

utterances are only likely to be said under particular circumstances. These may be related to where someone is spatially, the history of her previous interactions, how she is gesturing, what devices in the room are doing, etc. At the simplest level, this can range from the implausibility of someone saying "stop the video," when none is playing, to more complex dependencies, such as the meaninglessness of asking "What's the weather there?" if no geographic entity has somehow been brought to the room's attention.

The context-dependency of these grammars is not contained within the linguistic formalism itself, which allows us to use an extremely simple representation. Rather, the room's software agents are responsible for setting and modifying the activation states of grammars they create based on whatever information the room's other software agents can provide about current goings on in and state of the room. (See Figure 2.) For example, if the room starts showing a video clip, the agent that controls the showing of videos activates the grammars that involve VCR operation. When the clip stops, these grammars are in turn deactivated. More interesting cues can involve the location of someone inside the room. For example, the fact that someone has moved near an interactive displayed map is sufficient reason for the room to pay increased attention for spoken utterances involving geographic information. However, until that cue is received, it seems quite reasonable not only for the room to ignore such requests but to not recognize them at all especially given the error rate of current speech recognition technology. We note there may be cases where this is inappropriate; for example, the room might alternatively need to recognize out-of-context utterances in order to provide guidance to a user. We are investigating techniques for dealing with this, such as having the room iteratively broaden the set of active grammars and proactively offer assistance in case the user's speech is not being recognized.

When users shift to a new application context, the system lowers the relative rankings of and eventually deactivates grammars from the previous contexts according to a least-recently-used strategy. Thus, agents need not explicitly deactivate all of their grammars nor even know all appropriate circumstances for doing so.

Notions of context can also help adjust expected probabilities of utterances. Even in systems where all utterances are valid at all times, it is generally not the case that

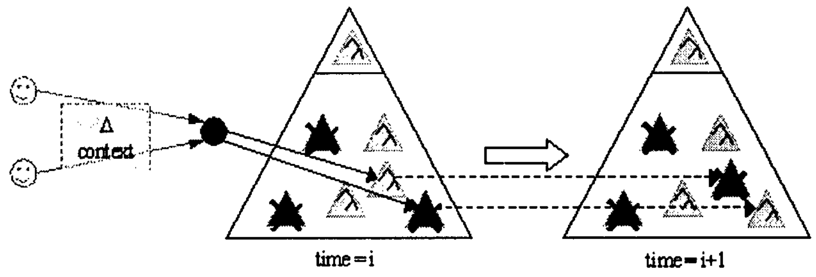


Fig. 2. A transition in the forest of grammars. An agent can activate and deactivate its grammars based on context changes in the Intelligent Room. Notification of context changes comes in the form of messages from other room agents.

all utterances are equally likely at all times. For example, tracking context can help disambiguate the output of bigram-based speech recognition systems that return the probabilistically weighted N-best set of utterances for each recognition event. We used this scheme to process the results returned by the Galaxy System [14], which was the first speech recognition system used in the early days of the Intelligent Room.

We have found it useful to have several different notions of ongoing room context for determining which grammars are active at any given moment. However, no single agent defines "the context" but rather the context is a product of many loosely connected entities. In ranked order, these consist of the following:

1. Always active grammars – These are for low-level control and providing feedback to the room. These grammars allow direct manipulation of room state; we have found it essential for users to feel they control the room's physical infrastructure if they are to feel comfortable interacting with it. These grammars also allow the manual adjustment of various room parameters in the event of incorrect data from one of the room's input modalities.
2. Context shifting grammars – These are explicit cues for the room to change its context. They generally start new room activities and automatically lead to changes in the contents of the next two categories.
3. Current applications' grammars — These are application specific verbal interactions with the room given its current state. The room frequently modifies these grammars while it is running.
4. Previous applications' grammars — These are for interacting with previously run applications. These are particularly useful in case of inadvertent or incorrect context shift or if some device failed to respond appropriately and must be corrected via verbal interaction. Backgrounded tasks often have grammars here so they can be quickly recalled and resumed.

Agents can also modify the structure and content of an extant grammar. This ability is used currently only for inserting and deleting noun phrases to reflect newly obtained information. This can be gotten from: the user verbally dictating new phrases to the room; mechanical extraction from other sources, (e.g., anchor link text in web pages); or the room applying machine learning techniques to augment its vocabulary (as discussed below).

Another advantage of the distributed nature of the grammars is individual agents can monitor their small piece of the overall context in a very simplistic fashion. For example, the VCR agent can pay attention to events only relevant to knowing whether the VCR is part of the context or not, and modify its associated grammars accordingly. This avoids the problem of clearly defining the overall context, and also eliminates the need for control logic for deciding which grammars should be active when.

Underlying Speech Technology and Computational Complexity

For processing spoken utterances, we use IBM's ViaVoice speech recognition system. ViaVoice is a commercially available system primarily used for continuous speech dictation. This, with its relatively low word accuracy for single word and short utterances, would have been an intolerable speech interface to the room. However, ViaVoice also supports explicit construction of continuous speech, context-free recognition grammars, which allow for much higher degrees of recognition accuracy. Via its Java interface, it also provides control over low-level aspects of its behavior to external applications, which makes it ideal for incorporating into other systems.

We wanted the Intelligent Room's recognition grammars to be reasonably unconstrained. In particular, we wanted to allow people to interact with the room without memorizing scripts of recognized utterances or lists of permissible syntactic constructions. However, we found that as our grammars became increasingly large, speech recognition accuracy fell correspondingly. As room grammars started to support more than a few thousand individual utterances, recognition accuracy dropped below acceptable levels.

There is a clear tradeoff between making the room's recognition grammars sufficiently large so that people can express themselves somewhat freely versus making the grammars small enough so that the system runs with high accuracy and in real-time. Thus, we decided to make use of the natural context specificity in room applications so that agents could dynamically activate different subsets of grammars depending on the context of the activity within the Intelligent Room.

5 Sample Interactions

The Intelligent Room supports a variety of narrow application domains, all of which can run simultaneously. The collection of all these domains in turn gives an extremely broad and flexible 'domain' encompassing a wide range of tasks. The distributed, independent control of the recognition forest allows for this—there is no need for a centralized controller. The room's applications range from simple voice control over physical devices to more complex multimodal scenarios involving position, gesture tracking and spoken dialog. We first outline these domains and then present two applications in more detail:

1. Manual control over devices – These include the Intelligent Room's lights, blinds, drapes, VCRs, video displays, stereo components, etc.

2. Manual interaction with modal subsystems – We have found it extremely useful to have direct verbal interactions with the room's modalities. These can be used to gather information about what the room is observing, to modify internal representations of its state, or to correct a perceptual error. It is also of enormous benefit to be able to verbally interact with the room's vision systems while developing or debugging them, because it is generally impossible to manually interact with them at a workstation while remaining in the camera's foveal areas.

3. Information access – There are many types of these interactions, including web browsing, weather reporting, accessing an online video collection, querying Haystack (a personal information manager), and the information retrieval system described below. The room also functions as a spoken language front-end to START, a natural language query database [9].

4. Presentation manager – This allows the Intelligent Room to assist in multimedia presentations and is a demonstration of the room's information management capabilities. A

lab tour guide agent that uses this application for presenting a broad overview of our laboratory's research to visitors has been previously described in [5].

5. Command post – This application provides the means to test full integration of all our modal subsystems and to experiment with different techniques for performing multimodal reconciliation. It is a mock command center for planning hurricane disaster relief.

We now present two primarily linguistic interactions with the Intelligent Room. They are annotated with the changes made to the recognition forest to reflect the course of ongoing interactions. The first of these is the above-mentioned command post. It makes use of two interactive projected displays that respond to finger pointing gestures. The second interaction is a primarily unimodal dialog that allows users to interactively refine a document retrieval query.

In the following interaction the user is attempting to plan disaster relief for a hurricane in the Virgin Islands.

Command Post:

User: *"Computer, stay awake."*
[The room will now listen for utterances without requiring they be prefaced by the word Computer.]
[The person's approach of projected displays causes the room to pay attention to statements involving them. This is illustrated in Figure 3.]
User: *"Show me the Virgin Islands."*
Room: "I'm showing the map on the display next to you."
[Room shows map on video display closest to the user.]
[Room activates grammars associated with the map.]
[User now points with his finger at St. Thomas.]
[Room adds nouns (such as city names) relating to St. Thomas to active grammars.]
[Room now considers St. Thomas the default geographical entity.]
User: *"Zoom in."*
[Room zooms in to St. Thomas.]
User: *"How far away is Hurricane Marilyn?"*
Room: "The distance between Hurricane Marilyn and the city of Charlotte Amalie located in St. Thomas is 145 miles."
User: *"Where's the nearest disaster field office?"*
[Room highlights them on the map.]
Room: "The St. Thomas DFO is located one mile outside of Charlotte Amalie. There is a new weather forecast available. Do you want to see it?"
[Room activates a grammar containing possible expected responses for this question.]
User: "Yes, show me the satellite image."
[Room displays image on adjacent projected display.]
[Additional weather grammars are now activated.]

The following interaction is an example of the user dynamically switching contexts while in the midst of interacting with the Intelligent Room. The room has an information retrieval system that helps users refine their queries by asking them questions that maximally disambiguate it. These questions are determined by searching a document tree obtained from the Alta Vista web search engine for phrases with the highest entropy levels.

Information retrieval:

User: *"Computer, I need information."*
[Because user didn't specify a topic, room will ask for it to be dictated.]
Room: "About what? Please tell me."
User: *"Software agents."*
Room: "Is software agents correct?"
User: *"Yes."*
[Room will now remember this phrase in the context of information retrieval to allow recognition of it in the future. Other agents might eavesdrop and also add this word for their own purposes, in an attempt to anticipate the user's eventual needs.]
Room: "Thank you. Please wait a moment...Are you interested in HCI, Research & Indexes, or General AI?"
[Room activates a grammar of expected responses based on these topics.]
User: *"I'm interested in HCI."*
[User can respond with continuous speech even though the phrase "H C I" was not in any room grammar before this interaction, since it was just added.]
[Above constructed response grammar is deactivated.]
Room: "Okay, are any of these documents of interest? I will put them on the left display."
[Room displays document list on wall.]
[Room activates a response grammar that incorporates the titles of displayed documents.]
User: *"Please move them to the display near the window."*
[Room does so]

The above examples illustrate how the room manages the recognition forest via data from both its perception subsystems and from expectations of what the user is likely to say in a given situation. This allows the room to approximate natural linguistic interactions with the user. Of course, it is still somewhat stilted, and the user cannot make complete non-sequiturs. Overall, however, the main problem we have is convincing the user to push the limits and speak naturally, rather than linguistically downsizing and second-guessing the room's capabilities, but this fault is prevalent in most, if not all, current speech understanding systems.

6 Achieving Design Goals

We review the design goals presented in the first section, considering not only how well they were achieved, but possible remedies for where our approach was unsuccessful.

A language modality

A key aspect of the Intelligent Room is for an occupant to have full access to all of the room's computational power regardless of where in the room she is. She should not need to type at a particular keyboard nor interact with a particular display to interact with the room, and in fact, the room does not have a keyboard or mouse within it.

Therefore, we decided that in many circumstances speech interactions had to be unimodal. It could not be the case that the room would need to display candidates for recognized utterances, thereby allowing the user to select among them or to disambiguate didactic references. This is contrasted with the approach of [11,14] where users provide critical feedback during the recognition process via a graphical user interface. Our approach of having the room ask the user when it is unsure of something can be somewhat intrusive, but it is certainly no more so than a graphical interface.

Context sensitivity

We wanted to make use of context in terms of both the room and user's states to be able to both resolve diectic references and control sets of possible utterances and who should receive those utterances. As in most speech understanding efforts, we wanted to support some measure of natural syntactic variability on the part of a person interacting with room. Our intent was to leverage off the well-defined notions of context inherent in the Intelligent Room's application domains to keep the total active grammar size small at any given time. This can be enormously frustrating if the room inappropriately deactivates a grammar to which the user would still like to refer. We are currently exploring techniques for dealing with this, such as reprocessing the spoken audio signal under an iteratively broadened set of grammars.

Non-static recognition sets

We sought to avoid limiting the room to a static set of recognition grammars. It would not have been reasonable to suppose that we could determine everything in advance users would want to say, and it would have made routine tasks like information retrieval difficult, if not impossible.

The ability of agents to change grammars on the fly has proved to be extremely useful, in applications such ranging from web browsing, where link anchor text is captured, to information retrieval, which typically involves an iterative query refinement process. The ability of the room to incorporate user-dictated noun-phrases into its recognition grammars is one of the capabilities that most impresses new users.

On a larger scale, adding agents should be easy. By having individual agents control their own grammars and activation states, agents can indeed be added quickly and without worry as to their disrupting other agents' interfaces.

Simplicity

Finally, we were also interested in employing very shallow linguistic knowledge during implementation to minimize the knowledge engineering problem. Given that new room applications are being created on a regular basis, it is not possible to build carefully handcrafted linguistic models of expected input. Speech orientated agents have proliferated markedly since our system came up, to a point of fault. Speech is such a natural and easy modality that the temptation to solve all problems with it has distracted us from really striving for a multi-modal system that pays attention to harder to discern inputs, such as gesture.

Future work on the system includes incorporating a machine learning mechanism into the recognition forest so that it can learn the probabilities of individual grammars be used in particular application contexts. We are also interested in learning the transition probabilities among the grammars, to better predict activation states without requiring explicit action be taken by the room's software agents.

8 References

[1] Andry, F., Fraser, N.M., McGlashan, S., Thornton, S., and Youd, J. Making DATR Work for Speech: Lexicon Compilation in SUNDIAL. Computational Linguistics, 18(3), pp. 245-267. 1992.

[2] Bobick, A.; Intille, S.; Davis, J.; Baird, F.; Pinhanez, C.; Campbell, L.; Ivanov, Y.; Schütte, A.; and Wilson, A. Design Decisions for Interactive Environments: Evaluating the KidsRoom. Proceedings of the 1998 AAAI Spring Symposium on Intelligent Environments. AAAI TR SS-98-02. 1998.

[3] Chung, G., and Seneff, S. Improvements in Speech Understanding Accuracy Through the Integration of Hierarchical Linguistic, Prosodic and Phonological Constraints in the Jupiter Domain, Proc. ICSLP '98, November 1998.

[4] Coen, M. A Prototype Intelligent Environment. In Streitz, N., et al. (Eds.), Cooperative Buildings - Integrating Information, Organization, and Architecture. Proceedings of the First International Workshop on Cooperative Buildings (CoBuild'98). Lecture Notes in Computer Science. Springer: Heidelberg. 1998.

[5] Coen, M. Building Brains for Rooms: Designing Distributed Software Agents. In Proceedings of the Ninth Conference on Innovative Applications of Artificial Intelligence. (IAAI97). Providence, R.I. 1997.

[6] Coen, M. Design Principles for Intelligent Environments. In Proceedings of The Fifteenth National Conference on Artificial Intelligence. (AAAI98). Madison, Wisconsin. 1998.

[7] Coen, M. (ed.) Proceedings of the 1998 AAAI Spring Symposium on Intelligent Environments. AAAI TR SS-98-02. 1998.

[8] Federico, M. and Vernesoni, F. "A speech understanding architecture for an information query system". Proceedings of EUROSPEECH 95, Madrid, Spain, 1995.

[9] Katz, B.. Using English for Indexing and Retrieving. In Artificial Intelligence at MIT: Expanding Frontiers. Winston, P.; and Shellard, S. (editors). MIT Press, Cambridge, MA. Volume 1. 1990.

[10] Mozer, M. The Neural Network House: An Environment that Adapts to its Inhabitants. Proceedings of the AAAI 1998 Spring Symposium on Intelligent Environments. AAAI TR SS-98-02. 1998.

[11] Stock, O., Carenini, G., Cecconi, F., Franconi, E., Lavelli, Magnini, B., Pianesi, F., Ponzi, M., Samek-Lodovici, V., and Strapparava, C.. ALFRESCO: Enjoying the Combination of Natural Language Processing and Hypermedia for Information Exploration. In Maybury, M. (ed.), Intelligent Multimedia Interfaces, The MIT Press, pp. 197-224.

[12] Winograd, T. Five lectures on artificial intelligence. Technical Report, Stanford-CS, Number AI Memo 246, Stanford University, 1974.

[13] Woods, W., Kaplan, R., and Nash-Weber, B. The Lunar Sciences Natural Language Information System: Final Report. Technical Report, Bolt Beranek and Newman, Number 2378, June 1972.

[14] Zue, V. Human Computer Interactions Using Language Based Technology. IEEE International Symposium on Speech, Image Processing & Neural Networks. Hong Kong. 1994

Ambient Telepresence: Colleague Awareness in Smart Environments

Hans-W. Gellersen and Michael Beigl

Telecooperation Office (TecO), University of Karlsruhe,
Vincenz-Prießnitz-Str. 1, D-76131 Karlsruhe, Germany
{michael,hwg}@teco.edu

Abstract. Ambient Telepresence is a method to support informal awareness in distributed collaborative work, and to promote a sense of presence of people who are in fact not co-located. In this approach, everyday things that people use are augmented with awareness technology, creating a smart environment in which information on background activity can be collected and interpreted. This information is transmitted to remote sites, where it is rendered for peripheral awareness, using ambient media. The approach is demonstrated with the MediaCup environment, in which coffee cups are augmented with sensor, processing, and communication to obtain some basic cues on what people do.

1 Introduction

Ambient Telepresence is a method using smart environments for providing informal awareness in distributed collaborative work. 'Smartness' of the environment is used in two ways: to obtain information on background activity in the local work place[1], and to present this information in a non-distracting way at a remote location. The information communicated between locations is interpreted by the smart environment; this is in contrast to conventional methods for informal awareness which generally use video-based techniques to provide remote people with a sense of what's going on. In [4] we recently introduced the notion of white box context vs. black box context to distinguish interpreted awareness information from information simply channeled to a remote site.

In the field of computer-supported collaborative work (CSCW), awareness support systems are designed to provide distributed people with similar kinds of cues as available in face-to-face settings. Cues such as whether a colleague appears to be very busy help to assess availability for interaction and guide the social coordination of collaborative work. The common approach to provide such cues over distance is to use video communication to let people follow up on what's going on at a remote site. Awareness systems based on this approach mediate video material but doe not extract

[1] Background activity as opposed to the collaborative work activities in the foreground of a group support systems.

cues or any kind of higher-level context: this is up to the human recipient, distracting them from other tasks, and potentially overloading them (and their desktop displays) with information. Besides video-based awareness, some systems use technology infrastructure to obtain very specific cues for collaboration. The ActiveBadge system for example provides a smart environment to track the whereabouts of people, a fundamental cue for initiation of contact and collaboration [13]. In this kind of system, the people's location is a white box context, understood by the support system which can use this knowledge to generate suitable target representations, such as graphical location maps and audio event notifications.

The ambient telepresence approach that we propose in this paper is based on a smart infrastructure to obtain a multiplicity of specific cues useful for informal awareness. The key ideas are:

- Tracking manipulation of the everyday things people use rather than tracking the people themselves
- Building smartness into the things that surround us rather than introducing new smart devices
- Collecting white box context rather than black box context to support informal awareness
- Mapping awareness information to a target representation that is not monopolizing attention and not competing for display resources

The paper is now organized as follows: we will first discuss related work on awareness support systems, on the use of physical devices for mediating awareness, and on ambient display of awareness information. In Section 3 we will introduce ambient telepresence, and in Section 4 we will describe the implementation of a system demonstrator. The demonstrator is based on MediaCups, ordinary coffee cups augmented with sensor technology, processing and communication to track their use.

2 Informal Awareness

In co-located teams, spontaneous meetings and casual interaction are used to coordinate the flow of work. Spontaneous meetings are facilitated by informal awareness of who is around and what they do. For distributed teams, real-time groupware for is readily available to provide communication channels for spontaneous casual interaction but as Cockburn and Greenberg point out, people still have trouble making contact for lack of mutual awareness [5]. In the CSCW community, this issue has been addressed with video-based methods for awareness support.

2. 1 Video-based Informal Awareness

Video-based methods for providing informal awareness include media spaces, video glances and snapshots. In *media spaces*, people can view remote offices and spaces through continuous video [1]. The use of continuous video raises issues of giving rather too much information, waisting bandwidth, and compromising privacy. In *video glances* the continuous video is replaced by a user-initiated brief two-way video connection to a remote person's location, which is like peeping into a colleague's office [12]. *Video snapshots*, on the other hand, provide for continuous awareness but replace the continuous video stream with snapshots of low resolution that are updated only every few minutes [10].

There are a number of problems associated with video-mediated informal awareness. Video provides only somewhat static and restricted views of a remote site, capturing awareness information only within limitations. Mediated video images are usually of low resolution or even deliberately blurred for the sake of privacy; this means the awareness information carried in the images is rather coarse. Further, people have to attend to their computer screens to obtain awareness information, and awareness displays compete with other applications for screen real estate. This latter issue was addressed in Buxton's Ubiquitous Media Spaces, using multiple videoconferencing units separate from the computer and placed in the architectural work space, each as surrogate for a remote person [3].

2. 2 Sensor-based Context-Awareness

An alternative to capturing awareness information with video is to use sensors. In contrast to video, sensors capture rather specific information from which can be interpreted and processed by computers. A simple example are infrared or radio frequency sensors that keep track of electronic badges worn by people, as in the ActiveBadge system [13]. Special-purpose sensors can be used to obtain other context relevant for informal awareness, for example to monitor whether the telephone is engaged, or whether the door is open or shut and [7].

Individual sensors capture only limited awareness information but it can be assumed that with the use of many sensors and the combination of different kinds of sensors a large degree of awareness can be achieved. Smart environments with ubiquitous sensors are in contrast to perceptual intelligence, shifting the cost for obtaining awareness from processing to infrastructure. Given the current advances in sensor technology with respect to size, cost and accuracy, their ubiquitous deployment in smart environments is becoming viable.

2.3 Things That Mediate

As alternative to the desktop-bound use of video, a number of systems demonstrate the use of physical devices separate from the computer to mediate awareness. In most of these systems, people interact rather explicitly through physical devices. Examples are Shaker [11] and inTouch [2] which facilitate interaction through pairs of haptic devices. In contrast there are only few examples in which physical devices are used for informal awareness of remote activity. Kuzuoka and Greenberg have designed a number of *Digital but Physical Surrogates* which are tangible representations of remote people [9]. These surrogates are used to indicate activity and availability of the people they represent. For example the peek-a-boo surrogate is a figurine that rotates to face away if the represented person becomes unavailable, as measured with simple sensors.

The use of physical devices in our approach, ambient telepresence, differs in two ways from Kuzuoka and Greenberg's work: ambient telepresence is based on computationally augmented everyday devices rather than newly introduced devices, and these devices are used to collect awareness information *while Physical but Digital Surrogates* are used to display the information. More directly related to ambient telepresence is the Internet Bed which is, however, rather one-of-a-kind installations. In the Internet Bed, the presence of a person in one bed is translated to warmth and heartbeat sounds on a remote bed [6].

2.4 Ambient display of awareness information

Ambient displays as investigated in the MIT MediaLab's AmbientRoom overcome limitations of conventional computer display and can be used to reflect activity in the information world in our surrounding physical environment [14]. Examples for ambient displays demonstrated in the AmbientRoom are water ripples, light patches and sounds. These displays exemplify calm technology and lend themselves to peripheral awareness, presenting information in the background rather than having it intrude in the foreground of other work activity.

In our ambient telepresence system, ambient displays are used to convey the feeling of a remote person's presence. This relates to one of the installation in the AmbientRoom, promoting a remote hamster's presence by representing it's activity in a hamster wheel in a physical vibrating object [8].

3 Ambient Telepresence

We work with a simple definition for ambient telepresence:

Ambient Telepresence is a method to give someone the feeling that someone else is present while they are actually not co-located

Ambient telepresence connects people at different locations through several steps as shown in figure 1, and as illustrated in figure 2. First, information on the background activity of a person is obtained in their local environment. The technical approach is

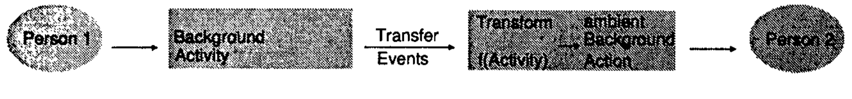


Fig. 1. Connecting locations for ambient telepresence: background activity sensed in one location is transmitted as event, and given an ambient representation at the other location.

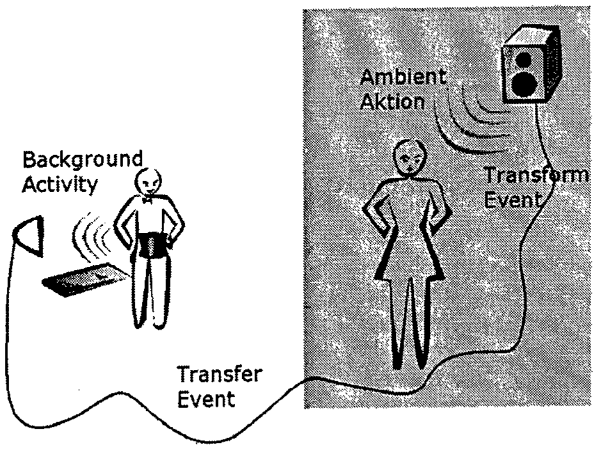


Fig. 2. Ambient Telepresence: a person's background activity is tracked, transferred as awareness event and transformed to an ambient representation at a remote location

to have sensors and processing embedded in the everyday things that people use in their environments. The collected information is interpreted to obtain context information at symbolic level, handled as events. The next step is to transfer these events to the remote location. At the remote site the event is processed by a dispatcher, mapping the event (i.e. the activity represented by the event) to an ambient display, available to a connected person for peripheral awareness. We assume a mapping to a representation that is perceived as natural, for example by generating sounds directly associated with the represented background activity.

Figure 3 depicts the system architecture. The remote sites are connected via the Internet to exchange events. Locally, different media and computationally augmented devices are connected in a smart environment. The devices in this environment broadcast awareness information, and a dispatcher is used to channel information to remote sites. The dispatcher is also used for dispatching events received from remote sites to local media for their representation.

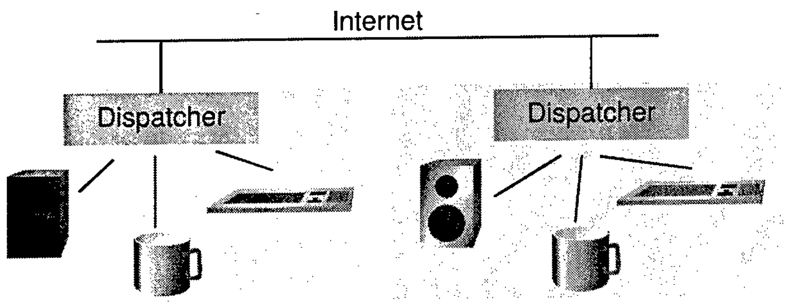


Fig. 3. System architecture for ambient telepresence: local devices are integrated in a smart environment controlled by a dispatcher, and remote sites are connected via the Internet.

4 Implementation and Demonstration

For demonstration of ambient telepresence we have implemented a setup as shown in figure 4. Background activity in the workplace is tracked by monitoring manipulation of computer keyboards and coffee cups. At a connected remote site sounds are generated to represent the remote activity, to give it a virtual presence.

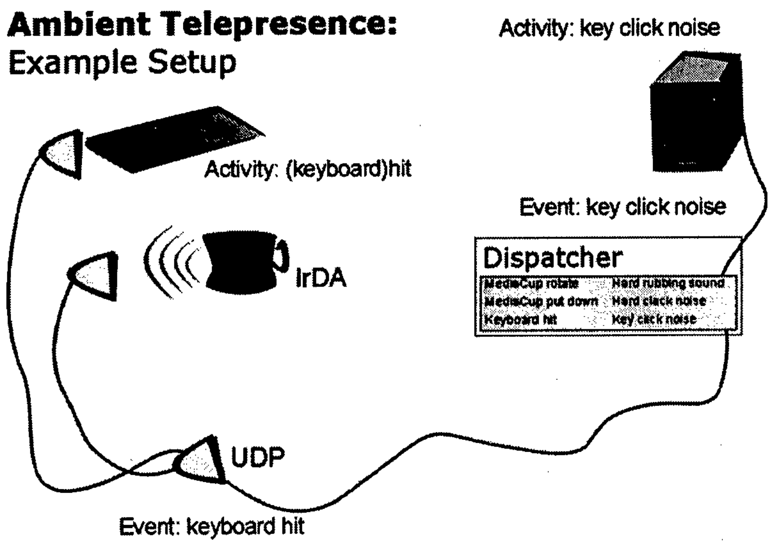


Fig. 4. Setup for demonstration of ambient telepresence

4. 1 A Smart Environment for Ambient Telepresence

In our office environment we have created a smart environment to demonstrate ambient telepresence. The backbone of this environment is an infrared network with a transceiver infrastructure mounted under the ceiling. We have used HP's HSDL 1001 IrDA Transceiver with 15 ° range, and about 1m^2 footprint. Transceivers are connected via serial line to a computer that connects the environment to the Internet, to connect with remote sites.

As example for augmentation of everyday objects with awareness technology we have developed MediaCups which are coffee cups with built-in sensors, processing and communication. The MediaCup hardware comprises sensors for temperature and acceleration, a PIC 16F84 microcontroller, an infrared diode for communication, and a standard Lithium battery (3V, 120mAh). To track how the cup is handled, we have integrated the two-axis acceleration sensor ADXL202AQC of Analog Devices, which can measure both dynamic and static acceleration. The sensor uses 0,6 mA and is turned off between measurement cycles to save power. For temperature sensing we have integrated the DS1621 Dallas Semiconductor chip measuring from –55 to +125 °C, with 1μA standby current, and 400μA communication current. The microcontroller has 1792 Byte Flash RAM for programs, 68 Byte RAM, and 13 I/O ports used for control of temperature chip, accelerometer, and infrared diode. With 4 MHz, power consumption is below 2mA, and in sleep mode below 1 μA. With the Lithium battery, the MediaCup can be powered for approximately 2-3 weeks.

Figure 5 shows two MediaCup prototypes. As shown on the left, the MediaCup hardware is embedded in a non-obtrusive way at the bottom of a coffee cup. The latest prototype shown on the right now has the hardware mounted in the rubber base of the HUC99 coffee cup, allowing removal so that the cup can be dish-washed. At present we have 8 of these cups operational in our office environment.

4.2 Capturing of Context

In the MediaCup, sensor readings are taken every 50ms for acceleration, and every 3 seconds for temperature. The raw sensor data is processed on the MediaCup, applying heuristics to obtain cues regarding handling and situation of the coffee cup. Acceleration sensor data is mapped to three distinct cues: cup is stationary, drinking out of the cup, and playing with the cup. Temperature data is mapped to the cues: filled up, cooled off, and actual temperature.

Cues are communicated every 15 seconds via a low-powered 3mm infrared sender SFH 409-s, using IrDA physical layer coding. MediaCups are tracked in the infrared transceiver network, so their location becomes available as additional context. The MediaCup can also communicate via transceivers already present in desktop and

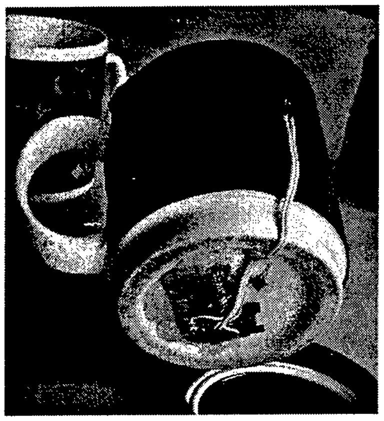


Fig. 5. MediaCup prototypes. Sensors, processor, and infrared diode are built into cup base.

laptop computers.

In addition to context obtained from the MediaCups we also collect awareness information available from people's interaction with their computers. To demonstrate this, we have included a monitor that keeps track of keyboard hits and of mouse manipulation.

4.3 Ambient Display of Context Information

In our current demonstrator we use audio only for ambient display of awareness information. The mapping of context to sounds is implemented in the media dispatcher. Table 1 lists a few examples for this mapping. The idea is basically to reproduce the noise associated with a certain activity. However, we have to note that our primary concern was technology demonstration, and that at this stage we abstracted from the available body of research on audio display.

MediaCup rotate	Hard rubbing sound
MediaCup put down	Hard clack noise
Keyboard hit	Key click noise

Table 1. Mapping awareness information to audio representation

5 Conclusion

We have introduced ambient telepresence as a new concept for support of informal awareness. The key ideas are: tracking of the everyday things people use to obtain information on background activity; augmentation of everyday objects with sensors and processing to facilitate their tracking; and use of ambient media to display obtained awareness information. To demonstrate the approach, a smart environment with computerized coffee cups, the MediaCups, was implemented. This environment was demonstrated at an exhibition and is now operational in the authors' work group.

We have used the MediaCup environment also for another colleague-awareness application [4]. In this application, context information obtained from the cups and from other sensors in the smart environment was used to support analysis of video streams for production a storyboard-like representation of recent activity in a workplace. This application as well as the discussed ambient telepresence demonstrate opportunities that smart environments create for support of collaborative work. However the presented work is only a first step to inform further research which most importantly will have to consider how these technical opportunities effect the people and their collaboration.

References

1. Bly, S.A., Harrison, S.R., Irwin, S., Media Spaces: *Video, Audio, and Computing*, Communications of the ACM, No.1, Vol.35, January (1993)
2. Brave, S., Ishii, H. and Dahley, D. Tangible bits for remote collaboration and communication. Proceedings of ACM Conference on Computer-Supported Collaborative Work (CSCW '98), 14-18 November, Seattle, USA, ACM Press New York 1998, p. 169-178.
3. Buxton, William A. S, *Ubiquitous Media and the Active Office*; Ubiquitous Video, Nikkei Electronics, 3.27 (no. 632), 187-195, (1995)
4. Chen, D., and Gellersen, H.-W. Recognition and Reasoning in an Awareness Support System for Generation of Storyboard-like Views of Recent Activity. *International Confcrence on Supporting Group Work (GROUP'99)*. November 14-17, Phoenix, USA, ACM Press New York, 1999.
5. Cockburn, A., and Greenberg, S. Making contact: Getting the group communicating with groupware. *Proceedings of ACM Conference on Organizational Computing Systems.* ACM Press New York, 1993, p. 31-41
6. Dodge, C. The Bed: A medium for intimate communication. *Proceedings of CHI'97 Extended Abstracts*, p. 371-372.
7. Ishii, H. and Ullmer, B., *Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms*, in Proceedings of Conference on Human Factors in Computing Systems (CHI'97), Atlanta, March 1997, ACM Press, p. 234-241
8. Ishii, H., Wisneski, C., Brave, S., Dahley, A., Gorbet, M., Ullmer, B., and Yarin, P. ambientROOM: Integrating Ambient Media with Architectural Space. *CHI'98 Video Program*, ACM.
9. Kuzuoka, H. and Greenberg, S. Mediating Awareness and Communication through Digital but Physical Surrogates. *ACM CHI'99 Video Proceedings and Extended Abstracts.*
10. Lee, A., Schlueter, K. and Girgensohn, A. NYNEX Portholes: Initial User Reactions and Redesign Implications. *Proceedings of International Conference on Supporting Group Work (GROUP'97).* November 1997, Phoenix, USA, ACM Press New York, p. 16-19.
11. Strong, R. and Gaver, B. Feather, Scent and Shaker: Supporting simple intimacy. *ACM CSCW'96 Poster Proceedings.*
12. Tang, J.C., Isaacs, E., and Rua, M. Supporting Distributed Groups with a Montage of Lightweight Interactions. *Proceedings of ACM Conference on Computer-Supported Cooperative Work (CSCW'94)*, p: 23-34, 1994.
13. Want R., Hopper A., Falcao V., and Gibbons J. The Active Badge Location System. *ACM Transactions on Information Systems*, Vol 10, No 1, 1992.
14. Wisneski, C., Ishii, H., Dahley, A., Gorbet, M., Brave, S., Ullmer, B. and Yarin, P., *Ambient Displays: Turning Architectual Space into an Interface between People and Digital Information*, in Proceedings of International Workshop on Cooperative Buildings, Darmstadt, Germany, Springer Press, pp. 22-32, February (1998)

INTELLIGENT ENVIRONMENTS

Intelligent Kinetic Systems in Architecture

Michael A. Fox , Bryant P. Yeh

Kinetic Design Group
Department of Architecture, Design Technology Room 10-461M
Massachusetts Institute of Technology, Cambridge MA 02139, USA
Email: mafox@mit.edu
Tel. (617) 252-1866
Fax. (617) 253-9407

Abstract. This research develops a concept for the application of smart environments to kinetic systems in architecture. The motivation lies in creating spaces and objects that can physically re-configure themselves to meet changing needs. Intelligent kinetic systems arise from the isomorphic convergence of three key elements: structural engineering, sensor technology and adaptable architecture. At the intersection of these areas exists an unexplored physical architecture tuned to address today's dynamic, flexible and constantly changing needs. Intelligent kinetic systems are unique to the field of architecture where objects are conventionally static, use is often singular, and responsive spatial adaptability is relatively unexplored.

Keywords. Adaptable Architecture, Responsive Architecture, Kinetic Structures, Kinematic Design, Intelligent Spaces

Introduction

This research develops a concept for the application smart environments to kinetic systems in architecture. The goal is to create flexible and responsively adaptable architectural spaces and objects. Novel applicational issues of smart environments arise through addressing how transformable objects can dynamically occupy predefined physical space as well as how moving physical objects can share a common physical space to create adaptable spatial configurations. We define kinetic architecture as buildings, or building components, with variable location or mobility and/or variable geometry or movement. Computer systems will interpret functional circumstances and direct the motor-controlled movements to change responsively and adaptively to better suit changing needs. Intelligent kinetic systems arise from the isomorphic convergence of three key elements: structural engineering, sensor technology and adaptable architecture.

Structural Engineering

Our concerns in structural engineering focus upon extending the possibilities of kinetic design. We address kinetic function as a technological design strategy for building types that are efficient in form, lightweight, and inherently flexible with respect to various contexts and a diversity of purposes. Facilitating adaptability, transportability, deployability, connectability and producability, they are ideally suited to accommodate and respond to changing needs. Recently manufacturing technologies have evolved to the degree where the creation of kinetic solutions can be both effectively and feasibly realized. Kinetic solutions are particularly suited to take advantage of technology, materials and techniques that exploit the potential of technological advance from other fields such as automotive, maritime, aviation, and the military. We classify kinetic systems into three main areas of research interest: Embedded, Deployable, and Dynamic kinetic structures.

Sensor Technology

This area addresses sensor technology as a computational control mechanism to accommodate and respond to changing needs. Systems will specifically be utilized to interpret functional circumstances and direct motor-controlled movements to change adaptively to better suit changing humans needs. We intend to exploit the case–specific advantages of both centralized and decentralized systems for the control of kinetic functions. Our development of such control mechanisms will draw upon previous research in AI called "Intelligent Environments" which is dedicated to creating spaces in which computation is seamlessly used to enhance ordinary activity. Many research areas in this field have achieved sufficient maturity to act as independent subsystems that can be beneficially incorporated into kinetic design. Our motivation lies is sensor technology as a means to actively controlling objects in the built environment in response to change.

Adaptable Architecture

An adaptable space flexibly responds to the requirements of any human activity from habitation, leisure, education, medicine, commerce and industry. Adaptability may range from multi-use interior re-organization to complete structure transformability to difficult site and programmatic response. Buildings that use fewer resources and that adapt efficiently to complex site and programmatic requirements, are particularly relevant to an industry increasingly aware of its environmental responsibilities. Adaptable architecture considers the rapidly changing patterns of human interaction with the built environment. New architectural types are emerging and evolving within today's technologically developing society. These new programs present practical architectural situations for unique and wholly unexplored applications that address today's dynamic, flexible and constantly changing activities.

Novel Applications in the Built Environment

Architectural applications for Responsive Kinematics arise from issues such as spatial efficiency, adaptability, shelter, security and transportability. Specific applications may include intelligent shading and acoustical devices, automobile-parking solutions, auditoriums, police box stations, teleconference stations, devices for ticketing and advertising, schools and pavilions, as well as flexible spaces such as sporting, convention and banquet facilities. Also of consideration are spaces with necessary fixed exterior configurations such as airplanes, boats, transport vehicles and automobiles. Through the application of intelligent kinetic systems, we can also explore how objects in the built environment might physically exist only when necessary and disappear or transform when they are not functionally necessary.

Although human technical prowess is embracing unprecedented sophistication, the permeation of this technology into architecture as built form remains in its infancy. We believe that in addition to addressing existing needs, intelligent kinetic systems will expose new programs and forms as this technology is incorporated into our everyday lives. An investigation into possible applications must consider the rapidly changing patterns of human interaction with the built environment. New architectural typologies are emerging and evolving within today's technologically developing society. These new programs present practical architectural situations where intelligently responsive kinetic solutions can be considered for unique and wholly unexplored applications. Intelligent kinetic systems are an approach for utilizing technology to create architecture that addresses today's dynamic, flexible and constantly changing activities.

Design Approach

We will describe some general mechanical and technological principals relevant to intelligently responsive kinetic design in architecture as categorized into three general research areas:

- Structural Innovation and Materials Advancement
- General Kinetic Typologies in Architecture
- Control Mechanisms

Structural Innovation and Materials Advancement

In developing such systems, the role of structure needs to be addressed not primarily or singularly, but rather as an integral component of a larger intelligently responsive kinetic system. The structural solutions shall consider in parallel both the *ways and means* for kinetic operability. *The ways* in which a kinetic structural solution performs may include among others, folding, sliding, expanding, and transforming in both size and shape. *The means* by which a kinetic structural solution performs may be, among others, pneumatic, chemical, magnetic, natural or mechanical.

Only recently, have manufacturing technologies evolved to the degree where the creation of intelligent kinetic architectural solutions can be both effectively and feasibly realized. Such systems are dependent upon both advanced computer control technology as well as the ability to manufacture high quality kinetic parts. New materials such as ceramics, polymers and gels, fabrics, metal compounds and composites are now available which can be integrated into intelligently responsive kinetic systems for exciting and novel applications. The integrative use of such materials in kinetic structures facilitates creative solutions in membrane, tensegrity, thermal, and acoustic systems.

General Kinetic Typologies in Architecture

We will classify kinetic structures in architecture into three general categorical areas:

Embedded Kinetic Structures

Embedded Kinetic structures are systems that exist within a larger architectural whole in a fixed location. The primary function is to control the larger architectural system or building, in response to changing factors. We draw upon an area of study within Active Control Research that focuses upon the design of structures to control the movements of a building through a system of tendons or moving masses tied to a feedback loop to sensors in the building. Changes are brought about by both environmental and human factors and may include axial, torsion, flexural, instability and vibration and sound. The engineer Guy Nordestrom indicates that if a building were built like a body, it could change it's posture tighten it's muscles and brace itself against the wind. As consequence, its structural mass could literally be cut in half.

Deployable Kinetic Structures

Deployable Kinetic structures typically exist in a temporary location and are easily transportable. Such systems possess the inherent capability to be constructed and deconstructed . Applications may include traveling exhibits, pavilions and self-assembling shelters in disaster areas. An example may be transportable public computer terminals, which can automate their own security.

Dynamic Kinetic Structures

Dynamic systems act independently with respect to the architectural whole. Applications may include louvers, doors, partitions, ceilings, walls and various modular components. An example may be an auditorium with ceiling configurations that can change dependent upon the audience and the performer locations for obtaining optimal acoustic properties. We will explore dynamic structures categorically as Mobile, Transformable and Incremental kinetic systems.

Control Mechanisms

We have defined kinetic in the context of architecture as the application of objects having mechanical parts that can be set in motion. Prior to describing the types of controlled movement for kinetic systems, we will list a general breakdown of the levels of machines (Zuk, 1967) by their ability to adapt to differing needs: 1) Singular in function 2) Multi-variable in function 3) Multi-variable in function with automatic control, and 4) Multi-variable in function with heuristic control. Within each of the defined typologies of kinetic structures: Embedded, Deployable and Dynamic, several levels of machines may exist simultaneously.

Types of Controlled Movement

To date we have developed numerous prototypical projects to study design and construction techniques, kinetic operability and maintenance, as well as issues of human and environmental interaction. Central to all of the projects is the means of controlling kinetic motion in architecture. We will illustrate the six general types below.

Internal Control

Systems in this category contain an internal control with respect to inherent constructional rotational and sliding constraints inherent. In this category falls architecture that is deployable and transportable. Such systems posses the potential for mechanical movement in a construction sense, yet they do not have any direct control device or mechanism. The "Folding Egg" is a prototype kinetic folding sheet structure shown in Fig. 1a. and Fig. 1b. demonstrating internal control. It is constructed from a low-cost recyclable material and forms a structurally stable collapsible three-dimensional truss structure. The Structure has a 5:1 folding ration and naturally „locks" into a stable open position. It can be constructed at a very low cost, with an R-value of 25 and a weight of less than 5.7-lbs/per sections.

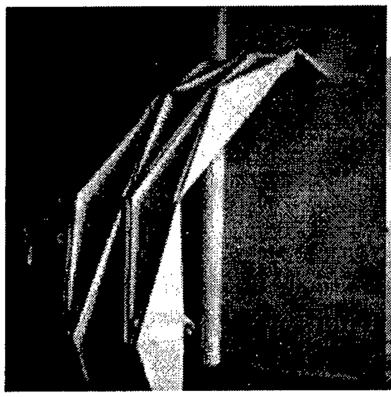
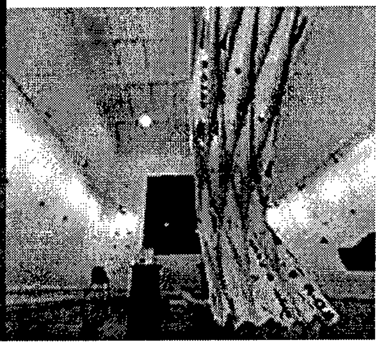



Fig. 1a. **Fig. 1b.**

Direct Control

Movement is actuated directly by any one of numerous energy sources including electrical motors, human energy or biomechanical change in response to environmental conditions. This skylight shown in Fig. 2a and Fig. 2b is one example of a simple rule based rotational system that can be applied to numerous geometrical configurations. Direct motor control actuates a 3-dimensional transformation resulting from one straight sliding motion. This skylight consists of eight glass panels held in compression against an aluminum frame. The glass is lined with an adjustable shading film to accommodate varying degrees of day lighting conditions.

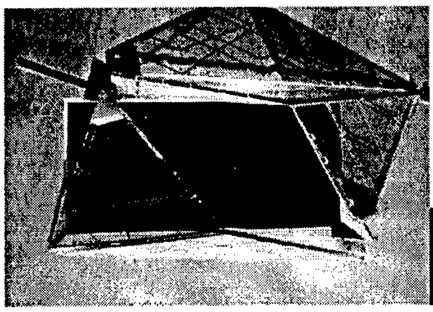


Fig. 2a.

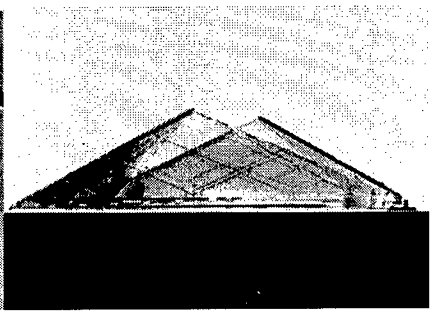


Fig. 2b.

In-Direct Control

Movement is actuated indirectly via a sensor feedback system. The basic system for control begins with an outside input to a sensor. The sensor must then relay a message to a control device. The control device relays an on/off operating instruction to an energy source for the actuation of movement. We define In-direct control here as a singular self-controlled response to a singular stimuli. The deployable teleconference station shown if Fig. 3a and Fig. 3b is a structure that houses a computer exhibit and teleconferencing station. The structure was designed to open automatically for use via a single motion sensor that triggers the deployment. When not in use, the object is closed into a simple secure (theft-proof) pyramid. While functioning, the structure transforms into a framed shell for communication. The structure was designed to express the conceptual aspects of a project (for the 1996 Lyon Biennale) in reference to language and communication as constantly transforming systems with multiple encapsulated meanings.

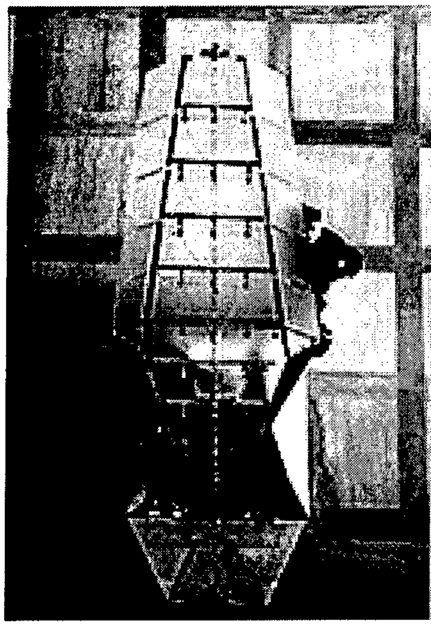


Fig. 3a.

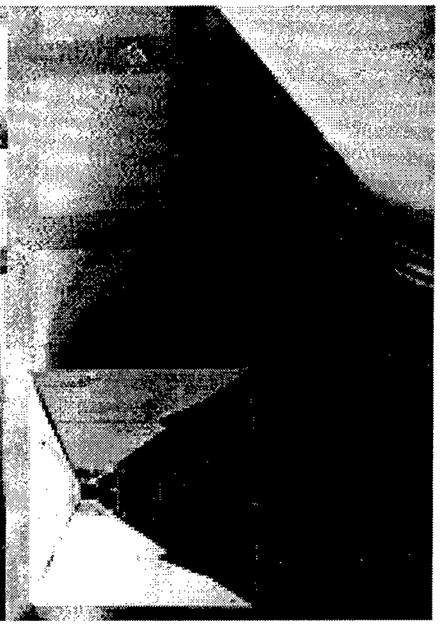


Fig. 3b.

Responsive In-Direct Control

The basic system of operation is the same as in In-Direct Control systems, however the control device may make decisions based on input form numerous sensors and make an optimized decision to send to the energy source for the actuation of movement for a singular object. The self-deployable macro-modular tent system shown in Fig. 4 can be combined into numerous structurally stable configurations. It has been conceptualized for a distributed sensor system to respond to natural day lighting conditions. Each vertical row of panels contains an individual sensor; as a vertical row deploys it pulls it's neighbor row of panels into a partially deployed position. The individual panels are supported on a steel structure made up of tie-rods, struts and columns. The membrane is made up of lightweight, semi-transparent anticlastic surfaces. The modular, lightweight units can be easily disassembled for transportation. A medium scale light-sensitive vault is currently under construction.

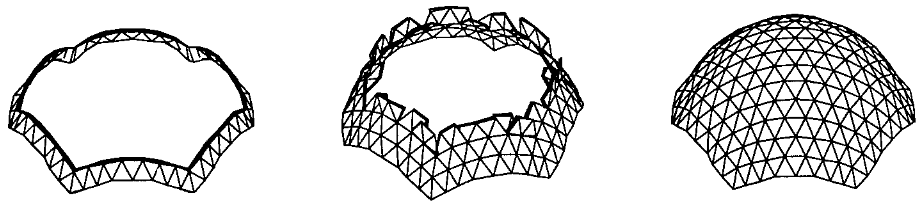


Fig. 4.

Ubiquitous Responsive In-Direct Control

Movement in this level is the result of many autonomous sensor/motor (actuator) pairs acting together as a networked whole. The control system necessitates a "feedback" control algorithm that is predictive and auto-adaptive. The Kinetic Wall shown in Fig. 6A and Fig. 6B is membrane or enclosure system that is at once both structure and envelope, both solid and plastic, a super-skin that could be used either as a temporary structure or incorporated into an existing structure. The structure is basically an assembly of one primary cell design. Each 'cell' is a grouping of three equal sized members that form a equilateral triangle. A larger hexagonal unit of comprised of six triangles grouped together around a central mast, which forms a 'panel' unit. To facilitate motion, a cable is strung along the back of the two vertical members formed by each cell. The cables begin at the top vertex of each hexagonal panel and run through the top of the panel's mast and then back down the adjacent vertical members of each preceding cell. This configuration allows individual control of each panel in relation to the adjacent panel. It also affords the ability to add more panels to create larger total surfaces. In this way a controllable dynamic surface skin is formed. Control is achieved by a series of computer-controlled servos that control each cable Three separate servomotors actuate each cable and are located below the base. The servomotors are actuated by a servo control device that is in turn controlled by a computer interface. Each servo controls the rotation of one panel. Activating all three servomechanisms in unison can create a controlled curvature. Integrated computer control is done with a system of positional sensor devices attached to each panel. Control is based on a feedback loop system. The system is mobile and can be controlled either by an active computer control system or by direct human movement.

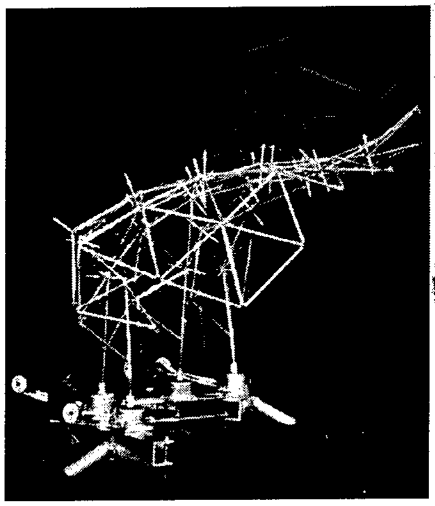


Fig. 5a.

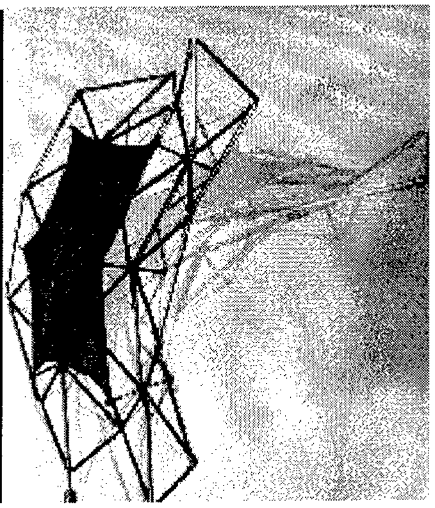


Fig. 5b.

Heuristic, Responsive In-Direct Control

Movement in this Level builds upon either singularly responsive or ubiquitously responsive self-adjusting movement. Such systems integrate a heuristic or learning capacity into the control mechanism. The systems learn through successful experiential adaptation to optimize a system in an environment in response to change. The Moderating Skylights shown if Fig. 6a and Fig. 6b, demonstrate a networked system of individual skylights that function together to optimize thermal and day lighting conditions. Each unit contains eight individual panels that slide along 4 straight lines towards the center of the panel to create an open position. The system maintains structural stability throughout all stages of deployment of the individual units. One of corner joints of a singular unit contains an individual cable attached to a servomotor that deploys the unit as an individual whole through sliding that joint towards the center of the unit. Integrated computer control is done with a system of positional sensor devices attached to each panel. Each panel further consists of photovoltaic cell paneling under which lies a layer of shading film/moisture barrier of variable self-adjusting opacity. This skin is affixed to a ribbed Plexiglas panel affixed to a structural aluminum frame. Optimum thermal and natural day lighting conditions can be achieved through the algorithmic balance between the individual deployment of the panel units and the individual opacity variances.

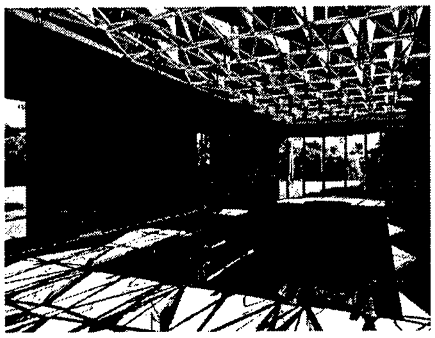
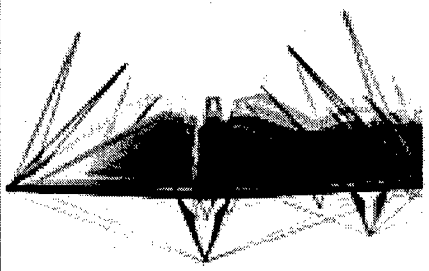



Fig. 6a. **Fig. 6b.**

When we look at the higher levels of computer controlled behaviors an interesting phenomenon can be observed with respect to actual physical built form with respect to all three of the illustrated types of kinetic structures: Embedded, Deployable and Dynamic. What we are describing is a structure as a mechanistic machine that is controlled by a separate non-mechanistic machine: the computer. Guy Nordenson describes the phenomenon as creating a building like a body: A system of bones and muscles and tendons and a brain that knows how to respond. In a building such as a skyscraper where the majority of the structural material is there to control the building during windstorms, a great deal of the structure would be rendered unnecessary under an intelligent static kinetic system. In deployable and dynamic systems as well, much of the structure will be reduced through the ability of a singular system to facilitate multi-uses via transformative adaptability. Buckminster fuller who coined it "Ephemeralization" first illustrated this concept of material reduction. Novel applications of smart environments arise both through addressing how transformable kinetic objects occupy predefined physical space as well as how moving physical objects can share a common physical space to create adaptable spatial configurations.

Conclusion

Intelligent kinetic systems arise from the isomorphic convergence of three key elements: structural engineering, sensor technology and adaptable architecture. At the intersection of these areas exists an unexplored architecture tuned to address today's dynamic, flexible and constantly changing needs. In developing a general concept for the application smart environments to kinetic systems in architecture, we introduce a new approach to architectural design, where objects are conventionally static, use is often singular, and responsive adaptability is typically unexplored

Only recently have computer-based control technologies and manufacturing technologies evolved to the degree where the creation of intelligent kinetic architectural solutions can be both effectively and feasibly realized.

Acknowledgments

This research is supported by the Graham Foundation and the Boston Architectural Educational Committee (EdCO).

References

1. Coen, M.: The Future Of Human-Computer Interaction or How I learned to stop worrying and love My Intelligent Room. To Appear, IEEE Intelligent Systems. 1999.
2. Coen, M.: Building Brains For Rooms: Designing Distributed Software Agents. American Association for Artificial Intelligence. (1997)
3. Fox, M. A.: Novel Affordances of Computation to the Design Process of Kinetic Structures, Thesis M.S., Massachusetts Institute of Technology, Cambridge, MA (1996)
4. Parsons, R. V.: Computer Aided Synthesis of Kinematic Linkages, Thesis M.S., Massachusetts Institute of Technology, Cambridge, MA (1996)

5. Kronenburg, R.: Transportable Environments: Papers from the International Conference on Portable Architecture, E & FN Spon, London (1997)
6. Kronenburg, R.: Portable Architecture, Architectural Press, Oxford (1996)
7. Yeh, B.: Kinetic Wall, Thesis M.S., Massachusetts Institute of Technology, Cambridge, MA (1996)
8. Zuk, W. and Clark, Roger, H.: Kinetic Architecture. Van Nostrand Reinhold, New York (1970)
9. Zuk, W.: New Technologies: New Architecture. Van Nostrand Reinhold, New York (1995)

SmartOffice: An Intelligent and Interactive Environment

Christophe Le Gal, Jérôme Martin, and Guillaume Durand

Projet PRIMA — Lab. GRAVIR-IMAG
INRIA Rhône-Alpes
655, avenue de l'Europe
38330 - Montbonnot Saint Martin, France
e-mail: Christophe.Le-Gal@imag.fr

Abstract. This paper presents our Intelligent Environment called *SmartOffice*. In the *SmartOffice* the user can work as in a normal office. The office's intelligence observes the user in order to anticipate his intentions and augments his environment to communicate useful information. Computers are involved in user activities in order to help in everyday tasks. The system interacts with users using voice, gesture or movement.
The *SmartOffice* provides a test-bed for collaboration and combinaition of independant modules integrated into a single coherent application. Integration requires a flexible working environment in which module developpers should not worry about low-level communication between modules. This paper presents a flexible resource-oriented integration protocol, which we argue is necessary to build such an environment. All modules need not be aware which resources can be provided by each module. They communicate with the supervisor which acts as a resource-server. The supervisor is programmed using a rule-based language, in which the addition or the suppression of a module requires only the suppression of the corresponding rule.
Two major modules of the *SmartOffice* are considered to illustrate de role of the supervisor: The first module is the *MagicBoard*, an ordinary white board augmented by a camera and a video-projector.
To guess the user-intentions, the system must constantly be aware of the location of the users in the *SmartOffice*. Therefore we need a user localization module. This is the second module that we describe. An example of the tracking architecture is presented.

1 Motivation and Background

Surprisingly, some people claim that computers have introduced a reduction in the effectiveness of work. For those people, the arrival of computers simply means that a huge new box takes up space on their desk, and makes tasks that were simple more complicated. Adams' axiom humorously claims that computers increase the working speed by 100 % but increase the work load by 300 % [Add97]. Computers today are intrusive. Even if they claim to be, they are not truly user–friendly, since users must engage in an explicit oriented dialogue with the computer, instead of using more natural ways of interaction.

We believe that a computer should not involve a new way of working but simply augment the current working modes. Therefore, as pointed out by Weiser [Wei94], computers should be invisible, not demanding any adaptation from the user, while at the same time bringing him the benefits of data processing power. The idea of Coen [Coe98] is to *make computer–interfaces for people rather than people–interfaces for computers.*

Intelligent Environments (IE) aim to create environments that facilitate the use of computers, by being aware of the occupants and responding to their voice and gesture commands. The *SmartOffice* is an intelligent environment being developed at the GRAVIR–IMAG Lab. In the *SmartOffice* the user can work as in a normal (even computer–free) office. The office's intelligence observes the user in order to anticipate his intentions and augment the user's environment to communicate useful information.

As mentioned by Etzioni [Etz93], and applied in Intelligent Environments by Coen [Coe99], software environments (such as IE) provide a test–bed well–suited for techniques developed in the context of robotics. It would be a mistake to view an IE as a simulated robot, even though it is partially artificial. A more accurate view is to consider it a non simulated robot because it manages real sensors and effectors and also software applications. Consequently, as all robots, it needs control modules and a supervisor.

Coen [Coe97] proposed a distributed room control architecture based on communication between agents. Each agent is responsible for a specific room function. Agents can ask others to warn them when a particular event occurs. For example an agent can ask the speech recognition agent to be warned each time a specific utterance is spoken. The architecture proposed in this paper follows the same spirit. The main difference is that a supervisor is responsible for the communication between agents, thus acting as a resource–manager. All agents should not be aware of which resources can be provided by each agent. They should not have to ask *Tracker: give me what you know* but be able to pose questions such as "*Resource manager: give me the (x,y) position of the user*". As opposed to Coen's agent–oriented communication, we propose a resource–oriented communication.

Section 2 presents the material architecture of the *SmartOffice*. The *SmartOffice* is composed of fifty sensors (cameras and microphones) and three actuators, a video–projector and two speakers. A description of the supervisor is given in section 3. The supervisor controls numerous independent modules and makes them cooperate via exchange of push or pull messages. Sections 4 and 5 describes two major modules of the *SmartOffice*, the *MagicBoard* and the user–tracking procedure. The *MagicBoard* is the main "actuator" the *SmartOffice* can use to augment the user environment. The user–tracking module is a requisite for predicting user intentions. These applications demonstrates the communication between modules and the supervisor. Finally, in section 6 a discussion and a conclusion are given.

2 Material Architecture

The *SmartOffice* has been constructed in the INRIA Rhône–Alpes building. It measures 4.65 by 4.10 meters and a large office desk is found in the center. There are seven cameras, including five mobile ones, installed in the office. Four mobile cameras, located on the corners, are used for tracking one or more users. A wide angle camera observes the complete view of the office. It enables recognition of user activities and monitors tracking strategy. A mobile camera is also mounted on top of the screen in order to have a close–up view of the user in front of his computer. This view can be used for face recognition or facial expression recognition. Eight microphones spread out office ceiling are used for speech recognition. A white board is hooked onto the west wall where information are projected by a video–projector. The white board is also observed by a mobile camera. This camera is used for the *MagicBoard*, a module briefly described in a coming section. Figure 1 shows a drawing of the complete *SmartOffice* installation.

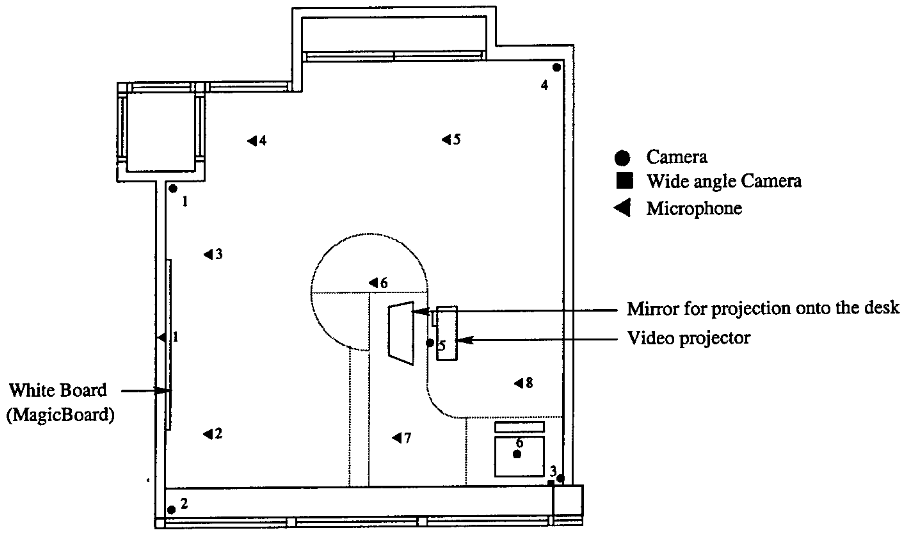


Fig. 1. The *SmartOffice* is an existing office equipped by seven cameras including five mobile, eight microphones, a video–projector and two speakers. This make possible to the system to "see" users gestures, "ear" voice and to give visual and audio feedback.

Six Linux Pentium III PCs connected with a fast Ethernet local network power the *SmartOffice*. Four computers are used to do vision processing i.e. user tracking and activities recognition. A fifth PC is dedicated to the *MagicBoard* for both video projection and image processing. A sixth computer hosts the supervisor and is linked to the Internet.

3 Supervisor

The software used in this application exists inside and outside of our group. Integrating these separate components into a coherent application requires a flexible supervisor. Therefore the supervisor is programmed using a rule–based language, in which the addition or the suppression of a module requires only the addition or the suppression of the corresponding rules, and has no influence on the other rules. The incrementality required for this research is therefore guaranteed.

The system architecture is centralized. All modules communicate with the supervisor through TCP/IP sockets. The distribution of the modules on several hosts is allowed. The modules are distributed in such a way that only the control–flow is transmitted via the Ethernet network; data–flow exists only between processes hosted on the same processor.

We distinguish two communication types: "push" messages and "pull" messages. In a push message, the writing process initiates the communication and interrupts the recipient process to give new facts. In a pull the recipient initiates the communication and receives the data from the writing process using a non–blocking read.

The communication protocol that we chose is a frame (attribute list) containing symbolic or numeric values for a set of attributes. For example, the tracking process associated with camera `1` informs the supervisor that the person identified by the token `1` was found at position `(1.32, 2.72)` and wears a `red` shirt using the following message:

```
[me: tracker, msg: pull,
 camera: 1, person: 1, x: 1.32, y: 2.72, shirt-color: red]
```

Some fields (`me` and `msg`) are mandatory. Nevertheless the frame is not statically defined and new fields can be added. For example, a new recognition modules is added during running. The recognition process is able to identify a person and informs the supervisor by pushing the message:

```
[me: recognition, msg: push,
 person: 2, name: "Suzanne", login: "suzy"]
```

Now, if an automatic login module is added, the supervisor is able to answer the request:

```
[me: login, msg: pull, want: login, person: 2]
```

This answer is possible without restarting the supervisor even if it was not conscious of the `login` field before.

A new process can dynamically add new rules to the supervisor. For example, if the *MagicBoard* module wants to be warned when user approached the board, it can send the following request to the supervisor:

```
[me: mboard, msg: rule,
 rule: 0.78 < x < 1.12 and y > 1.82
       => send mboard [msg: push, x: @x, y: @y]]
```

where **@x** refers to the value of x and not to the symbol x.

This architecture also allows the processing of redundant information thus increasing the global robustness of the system. Several modules can push the same field values to the supervisor. The supervisor use data–fusion algorithm to compute a more accurate value for this field. The following sections gives an example of this in the context of finger–tracking and user–localization.

4 The *MagicBoard*

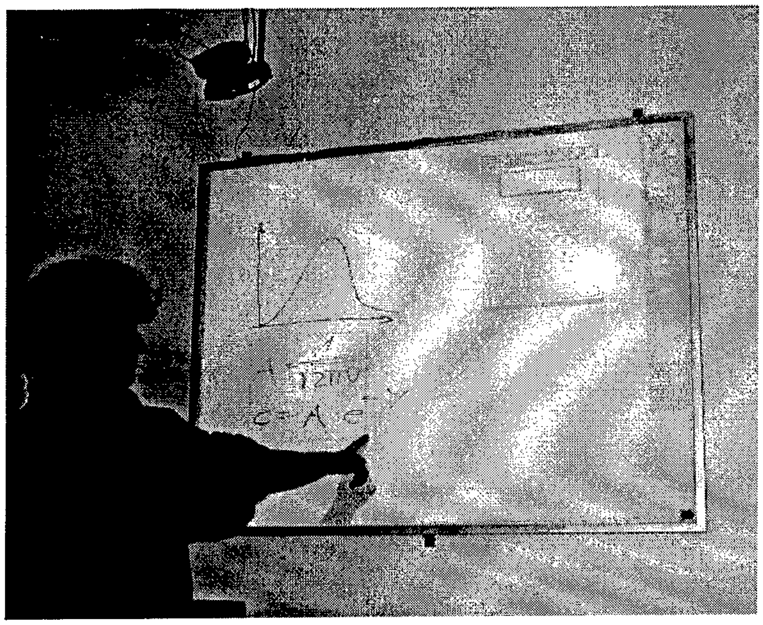


Fig. 2. The *MagicBoard* allows the combination of digital information and physical information on the board. The user can run augmented command for example copy–pasting of physical information using a gesture. An electronic calculator and a toolbar are projected onto the white board

The *MagicBoard* is an ordinary white board augmented with a camera and a video–projector. The video–projector allows the combination of digital information and physical information on the board. The camera digitalizes the physical informations present on the board and observes the user. The user can run augmented commands, for example printing, faxing or copy–pasting, by executing gestures. A rub out gesture cleans a region on the screen and a pointing gesture selects an option in a projected menu. Figure 4 shows an example where an user is copy–pasting physical formulas using a gesture. The projection of a electronic calculator is combined with physical formulas and drawing. Using the Internet, two or more *MagicBoards* can be combined to allow collaborative working.

The *MagicBoard*'s main components are a fast finger tracker, the X–Window system, and applications.

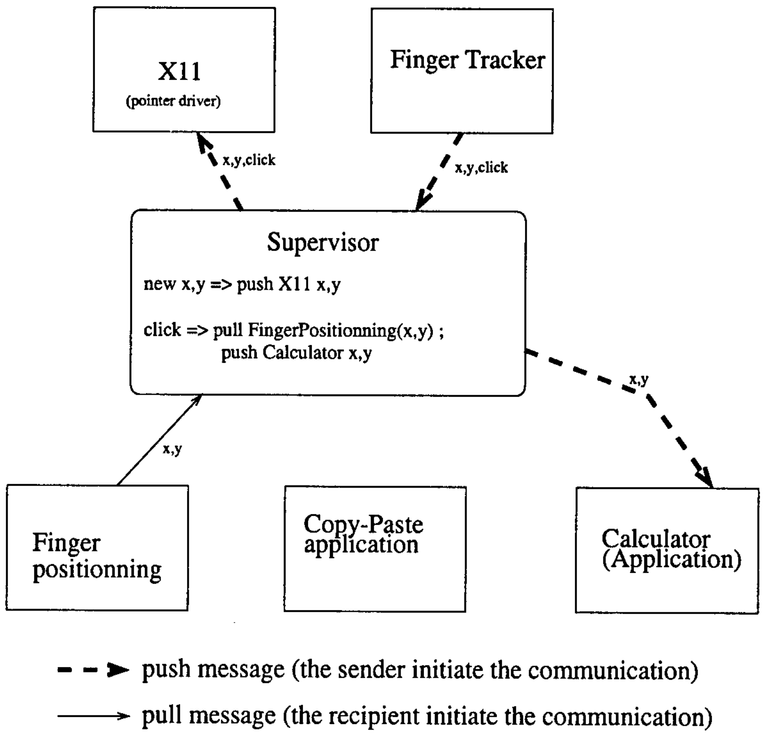


Fig. 3. The *MagicBoard* architecture. The X11 pointer driver is notified each time the finger tracker detects a finger motion or a click. The calculator application requests the supervisor to send to it an estimation of the finger position each time a click occurs. There is no direct link between the Copy–Paste application and the rest of the *MagicBoard* because the only information it needs (x,y) is transmitted via X11 protocol. Therefore, as far as architecture is concerned, the Copy–Paste application totally independent application from the supervisor

Finger Tracking System The finger tracking system is currently based on a rigid contour model described by Hall and Crowley [HC99]. The model consists of several points attached to the shape of the finger. The position and the orientation in a new frame are determined by selecting the transformation (translation and rotation) which maximizes a criterion which measures the correspondence between the shape of the finger in the image and the model.

Gestures are composed of three phases: *preparation*, *stroke* and *retraction* [Ken86]. During *preparation* and *retraction* phase, hand movements are fast. These phase do not have to be analyzed very precisely, whereas gestures need a high level of precision. This comment is especially true for pointing gestures. During the motion phase no accuracy is needed but a fast feedback is compulsory for comfortable use. During the selection, a more accurate localization is needed but speed is no longer crucial. Consequently each application can ask the supervisor for a more accurate (but slower) finger positioning, using the knowledge of what is projected onto the board.

***MagicBoard* Based on X–Windows** The tracker is used in such a way that it replaces the low level X11 pointer driver. The selection (i.e. the mouse click) is detected by a microphone behind the white board. Therefore the development of the *MagicBoard* modules has the same complexity as the development of other X11 applications. This allows development with high level tools such as Tcl/Tk. In addition, many existing X applications (xcalc, xfig, ...) can be effortlessly integrated into the *MagicBoard*. The *MagicBoard* environment is a simple window manager with the advantage that it can be configured in the same way as other window manager such as AfterStep.

Figure 4 shows an example of the *MagicBoard* architecture. More details about the *MagicBoard* are given by Hall et al [HLM+99].

5 User Localization

To guess the users intentions, the system must be constantly aware of the location of the users in the *SmartOffice*. When a person enters, he is referenced and an ID card is "pinned" on him. His height is estimated using the door frame. The ID card can be updated with more information on the person (e.g. his name or his shirt color).

A color based face tracker estimates the coordinates of the user in the camera image while the user is moving around the office.One tracker process is associated with each camera. Because several trackers are running simultaneously, several people can be tracked or one person can be tracked redundantly by several processes. Multiple estimations of the location of a single person are fused using a Kalman filter. Accurate localization is reinforced by a predictive model. The integration of a Kalman filter in such a supervisor was mostly inspired by SAVA project [CD94].

Cameras are calibrated using the office furniture, the position of which is known. Given the person's height and position, either standing or sitting, the previously calibrated camera allows the estimation of his location in the office by simple trigonometric computation. The estimation is an independent Gaussian probability for each camera tracker. Two user positions are considered: sitting or standing position. The activities recognition technique developed by Chomat and

Crowley [CC99] is used to detect the users sitting or in the process of standing up.

In case of a tracking failure, and at the beginning of the tracking processes, the tracker is initialized using a face–detector. The face–detector process is too slow to be used as a tracker. It can however be used to improve the robustness of the fast tracker: when an estimation of the face position is computed, the face–detector can "push" the estimation to the supervisor.

Figure 4 shows an example of the tracking architecture. In this example, two trackers are estimating one user location at 25 Hz using user position pulled from his ID card. The trackers are also reinforced by the face–detector running more slowly (about 5 Hz). Estimated locations are pushed to the supervisor. The Kalman filter pulls estimated locations and pushes fused locations. Changes in location are pushed to the graphic user interface display. *MagicBoard* is informed by the supervisor when user is approaching the board. When a standing up or sitting down event is detected, the activities detector pushes new user position to the supervisor. ID card is updated with the new position.

6 Discussion and Conclusion

The framework presented in this paper allows the efficient and easy integration of various modules. The integration is resource–oriented, i.e. the module programmer should not worry about which modules he should interact with, but only about which resources he needs. The available message types allow the building of system where no useless communication are performed. This the very first version of our system. Consequently much improvements are still to be done. The current version of the supervisor is mainly a resource manager. In future versions, the system will be able to perform more high–level operations.

Usage of expertise on the modules and planning techniques can automatically generate supervisor rules. For example if the supervisor is aware that

- *DoorObservation* module is able to compute a person height;
- Give a person height, *UserTracker* can know the person location

it will be able to automatically insert the rule *warn UserTracker whenever a person enters the office, and give it the height computed by DoorObservation.*

An unique supervisor can be a bottleneck when many independent modules are running simultaneously [Coe97]. In future versions several independent supervisors, each dedicated to specific tasks, will form a hierarchy of supervisors. For example the pointer position in the *MagicBoard* is not useful to the others modules in the *SmartOffice*. Consequently a separate supervisor could be instantiated to manage the *MagicBoard* and its applications. This supervisor would communicate only useful information to the main supervisor.

7 Acknowledgements

Development of the *SmartOffice* also involves the efforts of Daniela Hall, Karl Schwerdt, Olivier Chomat and Tomasz Kapuscinsky.

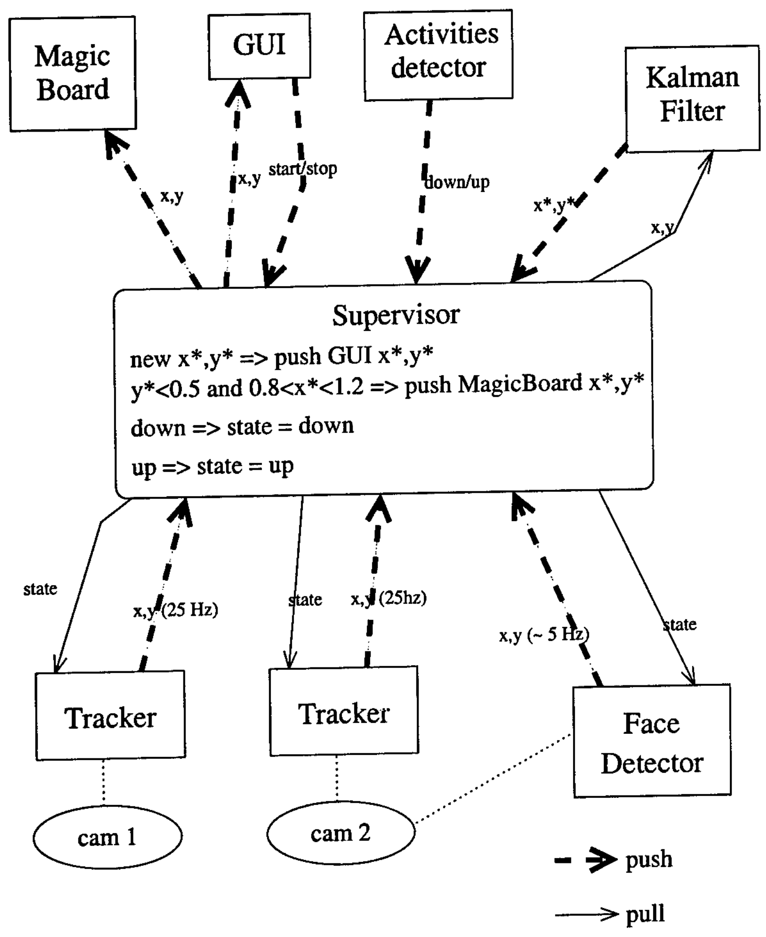


Fig. 4. The user–localization architecture with some sample rules. The GUI is notified of each movement of the user in order to update the display of the office configuration. The *MagicBoard* is notified only when the user enters a specific zone where he is supposed to use the *MagicBoard.* The supervisor is interrupted by the activities detector whenever the user raises or sits. The supervisor then updates a "state" value, which is constantly consulted by the trackers and the face–detector.

Nikla Duffy provides a valuable help for writing this paper.

Prof. James L. Crowley strongly supported these researches.

The combination of high–level programming (rules) with low level considerations (sockets), and more generally the integration of modules written in different languages is made easier by using the RAVI multilanguage platform developed by Prof. Augustin Lux et al [LZ97].

TMR Smart II financially sponsored this research.

More information can be found at `http://www-prima.imag.fr/SmartOffice`.

References

[Add97] S. Addams. *The Dilbert Future.* United Features Syndicate, Inc., 1997.

[CC99] O. Chomat and J.L. Crowley. Probabilistic recognition of activity using local appearance. In *Computer Vision and Pattern Recognition (CVPR'99)*, pages 104–109, April 1999.

[CD94] J.L. Crowley and C. Discours. The SAVA skeleton system. In J.L Crowley and H.I. Christensen, editors, *Vision as Process. Basic Research on Computer Vision Systems*, pages 23–45. Springer, 1994.

[Coe97] M. H. Coen. Building brains for rooms: Designing distributed software agents. In *Ninth Conference on Innovative Applications of Artificial Intelligence. (IAAI97)*, Providence, R.I., 1997.

[Coe98] M.H. Coen. Design principals for intelligent environments. In *Proceeding American Association for Artificial Intelligence 1998 Spring Symposium on Intelligent Environments*, Stanford, CA, USA, March 1998.

[Coe99] M. H. Coen. The future of human–computer interaction or how i learned to stop worrying and love my intelligent room. *IEEE Intelligent Systems*, 1999. to appears.

[Etz93] Oren Etzioni. Intelligence without robots: A reply to Brooks. *AI Magazine*, 14(4):7–13, 1993.

[HC99] D. Hall and J. L. Crowley. Tracking fingers and hands with a rigid contour model in an augmented reality. In *MANSE'99*, 1999. submission.

[HLM+99] D. Hall, C. Le Gal, J. Martin, O. Chomat, T. Kapuscinski, and J. L. Crowley. Magicboard: A contribution to an intelligent office environment. In *Proc. of the International Symposium on Intelligent Robotic Systems*, 1999.

[Ken86] A. Kendon. The biological foundations of gestures : Motor and semiotic aspects. In Nespoulous, Perron, and Lecours, editors, *Current Issues in the Study of Gesture.* Lawrence Erlbaum Associates, Hillsday, N.J., 1986.

[LZ97] A. Lux and B. Zoppis. An experimental multi-language environment for the development of intelligent robot systems. In *Proc. of the 4th International Symposium on Intelligent Robotic Systems*, 1997. Ravi is available at WWW `http://www-prima.imag.fr/Ravi`.

[Wei94] M. Weiser. The world is not a desktop. *Interactions*, pages 7–8, January 1994.

A Context-Based Infrastructure for Smart Environments

Anind K. Dey, Gregory D. Abowd and Daniel Salber

Graphics, Visualization and Usability Center and College of Computing,
Georgia Institute of Technology, Atlanta, GA, USA 30332-0280
{anind, abowd, salber}@cc.gatech.edu

Abstract. In order for a smart environment to provide services to its occupants, it must be able to detect its current state or *context* and determine what actions to take based on the context. We discuss the requirements for dealing with context in a smart environment and present a software infrastructure solution we have designed and implemented to help application designers build intelligent services and applications more easily. We describe the benefits of our infrastructure through applications that we have built.

1 Introduction

One of the goals of a smart environment is that it supports and enhances the abilities of its occupants in executing tasks. These tasks range from navigating through an unfamiliar space, to providing reminders for activities, to moving heavy objects for the elderly or disabled. In order to support the occupants, the smart environment must be able to both detect the current state or *context* in the environment and determine what actions to take based on this context information. We define context to be any information that can be used to characterize the situation of an entity, where an entity can be a person, place, or physical or computational object. This information can include physical gestures, relationship between the people and objects in the environment, features of the physical environment such as spatial layout and temperature, and identity and location of people and objects in the environment.

We define applications that use context to provide task-relevant information and/or services to a user to be *context-aware*. For example, a context-aware tour guide may use the user's location and interests to display relevant information to the user. A smart environment, therefore, will be populated by a collection of context-aware applications or services. The increased availability of commercial, off-the-shelf sensing technologies is making it more viable to sense context in a variety of environments. These sensing technologies enable smart environments to interpret and begin to understand the contextual cues of its occupants. The prevalence of powerful, networked computers makes it possible to use these technologies and distribute the context to multiple applications, in a somewhat ubiquitous fashion. So what has hindered applications from making greater use of context and from being context-aware?

A major problem has been the lack of uniform support for building and executing context-aware applications. Most context-aware applications have been built in an *ad* *hoc* manner, heavily influenced by the underlying technology used to acquire the context. This results in a lack of generality, requiring each new application to be built from the ground up. There has been little attempt to insulate the sensing of context in the physical world from using the context in applications. This makes it difficult to build new applications and to transfer existing applications to new environments with different or changing sensing technologies. To enable designers to easily build context-aware applications, there needs to be architectural support that provides the general mechanisms required by all context-aware applications. This support should include an insulation layer that on one side can hide the details of how context is sensed from applications, and on the other side, that can provide persistent context in a flexible manner without worrying about what, if any, applications require it.

At Georgia Tech, we have begun the construction of an experimental home to serve as a living laboratory for the experimentation of a number of ubiquitous computing research problems, including context-aware application development. One of the goals is to produce an "Aware Home" that will know contextual information about itself as well as information about its inhabitants [1][11].

There are a number of human-centered arguments for why an aware home is both desirable and worrisome. For instance, it may allow elderly to "age in place", remaining in familiar surroundings of their domicile as an alternative to more expensive and potentially alienating assisted living centers. But an environment that is instrumented to know too much about occupants' activities might not be a relaxing place to reside. Investigating these human-centered themes is a central focus of the long-term research plan. A necessary stepping-stone is the software infrastructure to allow us to rapidly prototype intelligent, or context-aware, services within such an environment.

This paper describes a distributed software infrastructure to support context-aware applications in the home environment. We provide a discussion of how context has been handled in previous work and why we would like to handle it as a generalized form of user input. In earlier work [17], we presented the concept of context widgets, which allow us to handle context in a manner analogous to user input. This paper discusses the infrastructure support necessary for using context and context widgets. We derive the requirements for the infrastructure by examining the differences between the uses of context and input. Next, we present our solution for supporting the design and execution of context-aware applications in smart environments. Finally, we discuss the benefits and limitations of the infrastructure, based on our experiences in building context-aware applications.

2 Discussion of Context Handling

We have discussed the importance of context in smart environments. The reason why context is not used more often is that there is no common way to acquire and handle context. In this section, we discuss how context has previously been acquired and

[1] More information on the Aware Home is available at http://www.cc.gatech.edu/fce/house

discuss some concepts that will allow us to handle context in the same manner as we handle user input.

2.1 Current Context Handling

In general, context is handled in an improvised fashion. Application developers choose the technique that is easiest to implement, at the expense of generality and reuse. We will now look at two common ways for handing context: connecting sensor drivers directly into applications and using servers to hide sensor details.

With some applications [8,16], the drivers for sensors used to detect context are directly hardwired into the applications themselves. Here, application designers are forced to write code that deals with the sensor details, using whatever protocol the sensors dictate. There are two problems with this technique. The first problem is that it makes the task of building a context-aware application very burdensome, by requiring application builders to deal with the potentially complex acquisition of context. The second problem with this technique is that it does not support good software engineering practices. The technique does not enforce separation of concerns between application semantics and the low-level details of context acquisition from individual sensors. This leads to a loss of generality, making the sensors difficult to reuse in other applications and difficult to use simultaneously in multiple applications.

The original Active Badge research took a slightly different approach [20]. Here, a server was designed to poll the Active Badge sensor network and maintain current location information. Servers like this abstract the details of the sensors from the application. Applications that use servers simply poll the servers for the context information that they collect. This technique addresses both of the problems outlined in the previous technique. It relieves developers from the burden of dealing with the individual sensor details. The use of servers separates the application semantics from the low-level sensor details, making it easier for application designers to build context-aware applications and allowing multiple applications to use a single server.

However, this technique has two additional problems. First, applications that use these servers must be proactive, requesting context information when needed via a polling mechanism. The onus is on the application to determine when there are changes to the context and when those changes are interesting. The second problem is that these servers are developed independently, for each sensor or sensor type. Each server maintains a different interface for an application to interact with. This requires the application to deal with each server in a different way, much like dealing with different sensors. This may affect an application's ability to separate application semantics from context acquisition.

2.2 Current Input Handling

Ideally, we would like to handle context in the same manner as we handle user input. User interface toolkits support application designers in handling input. They provide an important abstraction to enable designers to use input without worrying about how the input was collected.

This abstraction is called a widget, or an interactor. The widget abstraction provides many benefits. The widget abstraction has been used not only in standard keyboard and mouse computing, but also with pen and speech input [1], and with the unconventional input devices used in virtual reality [13]. It facilitates the separation of application semantics from low-level input handling details. For example, an application does not have to be modified if a pen is used for pointing rather than a mouse. It supports reuse by allowing multiple applications to create their own instances of a widget. It contains not only a polling mechanism but also possesses a notification, or callback, mechanism to allow applications to obtain input information as it occurs. Finally, in a given toolkit, all the widgets have a common external interface. This means that an application can treat all widgets in a similar fashion, not having to deal with differences between individual widgets.

2.3 Analogy of Input Handling to Context Handling

There have been previous systems which handle context in the same way that we handle input [2, 18]. These attempts used servers that support both a polling mechanism and a notification mechanism. The notification mechanism relieves an application from having to poll a server to determine when interesting changes occur. However, this previous work has suffered from the design of specialized servers, that result in the lack of a common interface across servers [2], forcing applications to deal with each server in a distinct way. This results in a minimal range of server types being used (e.g. only location [18]).

Previously, we demonstrated the application of the widget abstraction to context handling [17]. We showed that *context widgets* provided the same benefits as GUI widgets: separation of concerns, reuse, easy access to context data through polling and notification mechanisms and a common interface. Context widgets encapsulate a single piece of context and abstract away the details of how the context is sensed. We demonstrated their utility and value through some example applications.

The use of the widget abstraction is clearly a positive step towards facilitating the use of context in applications. However, there are differences in how context and user input are gathered and used, requiring a new infrastructure to support the context widget construct. The remainder of this paper will describe the requirements for this architecture and will describe our architectural solution.

3 Infrastructure Requirements

Applying input handling techniques to context is necessary to help application designers build context-aware applications more easily. But, it is not sufficient. This is due to the difference in characteristics between context and user input. The important differences are:

- the source of user input is a single machine, but context in a smart environment can come from many, distributed sources
- user input and context both require abstractions to separate the details of the sensing mechanisms, but context requires additional abstractions because it is often not in the form required by an application
- widgets that obtain user input belong to the application that instantiated them, but widgets that obtain context are independent from the applications that use them

While there are some applications that use user input that have similar characteristics to context, (groupware [14] and virtual environments [5] deal with distributed input and user modeling techniques [9] abstract input, for example), they are not the norm. Because of the differences between input and context, unique infrastructure support is required for handling context and context widgets. We will now derive the requirements for this infrastructure.

3.1 Distribution of context-sensing network

Traditional user input comes from the keyboard and mouse. These devices are connected directly to the computer they are being used with. When dealing with context in an instrumented smart environment, the devices used to sense context most likely are not attached to the same computer running the application. For example, an indoor infrared positioning system may consist of many infrared emitters and detectors in a building. The sensors must be physically distributed and cannot all be directly connected to a single machine. In addition, multiple applications may require use of that location information and these applications may run on multiple computing devices. As environments and computers are becoming more instrumented, more context can be sensed, but this context will be coming from multiple, distributed machines. Support for the distribution of context is our first high-level requirement.

The need for distribution has a clear implication on infrastructure design stemming from the heterogeneity of computing platforms and programming languages that can be used to both collect and use context. Unlike with user input widgets, the programming languages used by the application to communicate with context widgets and used by the widgets themselves, may not be the same. The infrastructure must support interoperability of context widgets and applications on heterogeneous platforms.

3.2 Abstraction: Interpretation and Aggregation

There is a need to extend the existing notification and polling mechanisms to allow applications to retrieve context from distributed computers in the same way that they retrieve input from local widgets. There may be multiple layers that context data goes through before it reaches an application, due to the need for additional abstraction. For example, an application wants to be notified when meetings occur. At the lowest level, location information is interpreted to determine where various users are and identity information is used to check co-location. At the next level, this information is combined with sound level information to determine if a meeting is taking place.

From an application designer's perspective, the use of these multiple layers must be transparent.

In order to support this transparency, context must often be interpreted before it can be used by an application. An application may not be interested in the low-level information, and may only want to know when a meeting starts. In order for the interpretation to be reusable by multiple applications, it needs to be provided by the infrastructure.

To facilitate the building of context-aware applications, our infrastructure must support the aggregation of context about entities in the environment. Our definition of context given earlier describes the need to collect context information about the relevant entities (people, places, and objects) in the environment. With only the context widget abstraction, an application must communicate with several different context widgets in order to collect the necessary context about an interesting entity. This has negative impacts on both maintainability and efficiency. Aggregation is an abstraction that allows an application to only communicate with one component for each entity that it is interested in.

3.3 Component Persistence and History

With most GUI applications, widgets are instantiated, controlled and used by only a single application. In contrast, our context-aware applications do not instantiate individual context widgets, but must be able to access existing ones, when they require. This leads to a requirement that context widgets must be executing independently from the applications that use them. This eases the programming burden on the application designer by not requiring her to maintain the context widgets, while allowing her to easily communicate with them. Because context widgets run independently of applications, there is a need for them to be persistent, available all the time. It is not known *a priori* when applications will require certain context information; consequently, context widgets must be running perpetually to allow applications to contact them when needed. Take the call-forwarding example from the Active Badge research [20]. When a phone call was received, an application tried to forward the call to the phone nearest the intended recipient. The application could not locate the user if the Badge server was not active.

A final requirement linked to the need for execution persistence is the desire to maintain historical information. User input widgets maintain little, if any, historical information. For example, a file selection dialog box keeps track of only the most recent files that have been selected and allows a user to select those easily. In general though, if a more complete history is required, it is left up to the application to implement it. In comparison, a context widget must maintain a history of all the context it obtains. A context widget may collect context when no applications are interested in that particular context information. Therefore, there are no applications available to store that context. However, there may be an application in the future that requires the history of that context. For example, an application may need the location history for a user, in order to predict his future location. For this reason, context widgets must store their context.

3.4 Requirements Summary

We have presented requirements for a software infrastructure that supports context-aware applications. To summarize, these requirements are:

- allow applications to access context information from distributed machines in the same way they access user input information from the local machine;
- support execution on different platforms and the use of different programming languages;
- support for the interpretation of context information;
- support for the aggregation of context information;
- support independence and persistence of context widgets; and
- support the storing of context history.

In the next section, we describe the context-based infrastructure that we have built to address these requirements.

4 Description of Infrastructure

Our infrastructure was designed to address the requirements from the previous section. We used an object-oriented approach in designing the infrastructure. The infrastructure consists of three main types of objects:

- Widget, implements the widget abstraction
- Server, responsible for aggregation of context
- Interpreter, responsible for interpretation of context

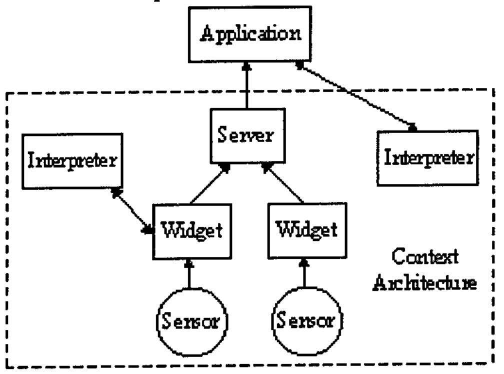


Figure 1. Data flow between applications and the context-based infrastructure.

Figure 1 shows the relationship between the objects and an application. Each of these objects is autonomous in execution. They are instantiated independently of each other and execute in their own threads, supporting our requirement for independence.

These objects can be instantiated all on a single or on multiple computing devices. Although our base implementation is written in Java, the mechanisms used are programming language independent, allowing implementations in other languages.

It is important to note that the infrastructure provides scaffolding for context-aware computing. By this we mean that it contains important abstractions and mechanisms for dealing with context, but it is not a complete solution, nor is it meant to be. The infrastructure supports building widgets and interpreters required by an application, but will not necessarily have them available. Compared to input, there are a larger variety of sensors used to sense context and a larger variety of context. This makes it very difficult to provide all possible combinations of widgets and interpreters.

4.1 Context Widgets

A context widget, as mentioned earlier, has much in common with a user interface widget. It is defined by its attributes and callbacks. Attributes are pieces of context that it makes available to other components via polling or subscribing. Callbacks represent the types of events that the widget can use to notify subscribing components. Other components can query the widget's attributes and callbacks, so they don't have to know the widget capabilities at design time. A context widget supports both the polling and notification mechanisms to allow components to retrieve current context information. It allows components to retrieve historical context information. The basic Widget object provides these services for context widgets that subclass it.

Creating a new widget is very simple. A widget designer has to specify what attributes and callbacks the widget has, provide the code to communicate with the sensor being used, and when new data from the sensor is available, call 2 methods: `sendToSubscribers()` and `store()`. The Widget class provides both of these methods. The first method validates the data against the current subscriptions. Each time it finds a match, it sends the relevant data to the subscribing component. For example, multiple applications have subscribed to a Meeting Widget with different callbacks, attributes, and conditions. When the widget obtains new meeting information it sends it to the appropriate subscribers.

The second method adds the data to persistent storage, allowing other components to retrieve historical context information. This addresses our requirement for the storage of context history. The Widget class provides a default implementation for persistent storage using MySQL, a freeware database. The persistent storage mechanism is "pluggable". A widget designer not wanting to use the default mechanism can provide a class that implements a temporary cache and allows the storage and retrieval of information from some persistent storage. The name of the class is given to the widget at run time, allowing the new storage mechanism to be used.

4.2 Context Servers

Context servers implement the aggregation abstraction, which is one of our requirements. They are used to collect all the context about a particular entity, such as a person, for example. They were created to ease the job of an application programmer. Instead of being forced to subscribe to every widget that could provide information about a person of interest, the application can simply communicate with a single object, that person's context server. The context server is responsible for subscribing to every widget of interest, and acts as a proxy to the application.

The Server class is subclassed from the Widget class, inheriting all the methods and properties of widgets. It can be thought of, then, as a compound widget. Just like widgets, it has attributes and callbacks, it can be subscribed to and polled, and its history can be retrieved. It differs in how the attributes and callbacks are determined. A server's attributes and callbacks are "inherited" from the widgets to which it has subscribed. When a server receives new data, it behaves like a widget and calls `store()` and `sendToSubscribers()`.

When a designer creates a new server, she simply has to provide the names of the widgets to subscribe to. In addition, she can provide any attributes or callbacks in addition to those of the widgets and a Conditions object. The Conditions object is used in each widget subscription, so the server only receives information it is interested in. For example, the Anind User Server would have the subscription condition that the name must equal "Anind".

4.3 Context Interpreters

Context interpreters are responsible for implementing the interpretation abstraction discussed in the requirements section. Interpretation of context has usually been performed by applications. By separating the interpretation abstraction from applications, we allow reuse of interpreters by multiple applications. An interpreter does not maintain any state information across individual interpretations, but when provided with state information, can interpret the information into another format or meaning. A simple example of an interpreter is one that converts a room location into a building location (e.g. Room 343 maps to Building A). A more complex example is one that takes location, identity and sound information and determines that a meeting is occurring. Context interpreters can be as simple or as complex as the designer wants.

Context to be interpreted is sent to an interpreter's `interpretData()` method. It returns the interpreted data to the component that called the interpreter. Interpreters can be called by widgets, servers, applications and even by other interpreters. When a designer creates a new interpreter, she only has to provide the following information: the incoming attributes, the outgoing attributes, and an implementation of `interpretData()`.

4.4 Communications Infrastructure

All of the top-level objects (widgets, servers, and interpreters) used are subclassed from a single object called BaseObject. The BaseObject class provides the basic communications infrastructure needed to communicate with the distributed components and abstract away the details of heterogeneous platforms and programming languages, supporting heterogeneity and distribution. Applications use this class to communicate with the context infrastructure. The communications includes dealing with both the passing of high-level data and low-level protocol details.

High-level Communications A basic communications infrastructure is needed to support semantic or high-level communications. BaseObject provides methods for communicating with the widgets, servers, and interpreters. In particular it facilitates subscribing and unsubscribing to, querying/polling and retrieving historical context

- Callback: the event of interest to the component
- Attributes: the particular widget attributes of interest
- Conditions: the conditions under which the widget should return data to the subscribing component

These three options essentially act together as a filter to control which data and under which conditions context events are sent from a widget to a subscribing component to be handled. This is an extension of the general subscription mechanism, where only callbacks can be specified. This helps to substantially reduce the amount of communication, which is important in a distributed infrastructure for performance reasons. This mechanism also makes it easier for application programmers to deal with context events, by delivering only the specific information the application is interested in.

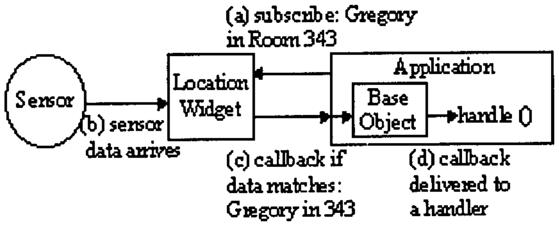

(a) subscribe: Gregory in Room 343
Sensor
(b) sensor data arrives
Location Widget
Application
Base Object
handle ()
(c) callback if data matches: Gregory in 343
(d) callback delivered to a handler

Figure 2. Example of application interacting with context widget. Arrows indicate communications flow.

When a widget sends callback data to a subscribing application, the application's BaseObject instance uses the data from the callback to determine which object to send the data to. It calls that object's `handle()` method, as shown in Figure 2. An application has subscribed (2a) to the Location Widget, wanting to know when Gregory has arrived in Room 343. When the Location Widget has new location information available (2b), it compares the information to see if the information meets the subscription conditions. If it does, it sends the information to the application (2c) and the application's BaseObject routes it to the `handle()` method (2d).

Communications Fundamentals The BaseObject acts as both a client and a server (in the client-server paradigm of distributed computing), sending and receiving communications. The BaseObject uses a "pluggable" communications scheme. By default, it supports communications using the HyperText Transfer Protocol (HTTP) for both sending and receiving messages. The language used for sending data is, by default, the eXtensible Markup Language (XML). The BaseObject does support the use of other communications protocols and data languages. When a designer wants an object to use a different protocol, SMTP (Simple Mail Transfer Protocol) for example, she creates an object that "speaks" the SMTP protocol for outgoing and incoming communications. She then passes the name of this object to the BaseObject at the time of instantiation. The BaseObject uses this object rather than the default object. In a similar fashion, a designer can replace XML with use his own data encoding scheme.

XML and HTTP were chosen for the default implementation because they support lightweight integration of distributed components and allow us to meet our requirement of support on heterogeneous platforms with multiple programming languages. XML is simply ASCII text and can be used on any platform or with any programming language that supports text parsing. HTTP requires TCP/IP and is ubiquitous in terms of platforms and programming languages that support it. Other alternatives to XML and HTTP include CORBA and RMI. Both were deemed too heavyweight, requiring additional components (an ORB or an RMI daemon) and in the case of RMI, would have forced us to use a specific programming language – Java.

5 Experience and Contributions

In this section, we describe the benefits and limitations of the context-based infrastructure for building context-aware applications. We will discuss these issues using context-aware applications that have already been created with this infrastructure.

We have built multiple types of each object (widgets, servers, and interpreters) and included this library of objects in our Context Toolkit[2]. Our default implementation of the infrastructure was developed in Java. However, the abstractions and mechanisms that address our requirements are both platform and programming language independent. To demonstrate this, we have also written applications and widgets in C++, Python, and Frontier. We have components of the infrastructure executing on Windows CE, 95 and NT, Linux, Solaris, IRIX, and the Macintosh platforms, including mobile, wearable, and desktop machines.

5.1 Benefits

The benefits of the context-based infrastructure are that it:
- hides the details of the context sensors; and
- supports lightweight integration of components;
- makes it easy to add context to existing applications that don't use context;
- makes it easy to add more context to applications that already use context; and
- supports reusability by multiple applications.

We will discuss these benefits, using the applications we have developed as examples.

Hides Context-Sensing Details A benefit of the infrastructure is that it hides details of the context sensors. This allows the underlying sensors to be replaced, without af-

[2] The Context Toolkit may be downloaded at http://www.cc.gatech.edu/fce/contexttoolkit/

fecting the application. We have built multiple applications that use location and identity context. In each case, when we change the sensor used to provide this context information, the applications do not have to be modified. We have built context widgets that sense location and identity around JavaRings (www.ibuttons.com), the PinPoint indoor positioning system (www.pinpointco.com), and the Texas Instruments TIRIS RFID system (www.ti.com). We have also built context widgets that determine activity context, using sound-level information, motion sensors, and door open/closed status. For a third type of context, we have built context widgets that can capture presentation information from a Web browser or PowerPoint.

Lightweight Integration The use of lightweight communications and integration mechanisms supports the deployment of the infrastructure and applications on multiple platforms using multiple programming languages. We have implemented an In/Out Board application [17] that keeps track of who is in our building and when they were last seen. This application has been implemented in a stand-alone mode in Java, and on the web[3] using Frontier on a Macintosh and Python on Linux. We built a notetaking application [7] that assists conference attendees in choosing which presentations they wanted to attend based on their personal interests and recommendations of colleagues and in taking notes on the presentations by automatically monitoring and capturing presentation information. The Conference Assistant application was executed on a Windows CE device, with widgets, servers, and interpreters executed in C++ and Java on Solaris and Windows NT and 95. A combination of motion sensors placed around entrances/exits to our research lab and magnetic reed switches that determine when a door is open/closed are being used to keep track of the number of people in our laboratory. This application was built in C++ and Java on Linux.

Simple to Use Context From an application builder's perspective, the infrastructure makes it simple to add the use of context to applications that previously did not use context. The DUMMBO [3] application is an augmented whiteboard that stores what is written on it and spoken around it, to aid in the capture of informal whiteboard meetings. The original version did not use context and began to capture audio and pen strokes when someone started writing on the whiteboard. The application was modified to initiate data capture when people were gathered around the whiteboard, but had not yet started writing. This enables capture of spoken information that would otherwise be lost. The change required the addition of 25 lines of Java code. The significant changes included 2 lines added to use the infrastructure, 1 line modified to enable the class to handle callbacks, and 17 lines that are application specific.

Simple to Add Context Based on these results, we argue that it is also simple to add additional context to applications that already use the infrastructure. In the Conference Assistant application, the use of context was made much simpler through the use of context servers (for each user and presentation room). From the application builder's viewpoint, the servers make accessing context information much easier than

[3] See http://fire.cc.gt.atl.ga.us/inout for the web version

dealing with the multiple widgets individually. If a new type of context was added to this application, the application could continue to use a context server as a proxy to the context information, requiring minimal changes.

Supports Multiple Simultaneous Applications An important feature of the infrastructure is independent execution. This lets the context-based infrastructure run independently of applications, which allows multiple applications to use the infrastructure simultaneously. We have built a number of applications that leverage off of the context-based infrastructure we have running in our lab, including the In/Out Board, a context-aware tour guide, the Conference Assistant, and a context-aware mailing list that only delivers mail to the current occupants of the building.

5.2 Current Limitations and Future Work

Although our infrastructure eases the building of context-aware applications, some limitations remain. In particular, it does not currently support continuous context, dealing with unreliable sensor data, transparently acquiring context from distributed components, and dealing with component failures. We discuss each of these issues and propose possible solutions.

Currently, the infrastructure only supports discrete context, and not continuous context such as a video feed or GPS location information. Given that a large number of sensors in a smart environment will provide continuous data, we need to support the collection of this type of context. We are investigating two possible solutions. The first is to provide a special mechanism that supports reliable, fast streaming of continuous context from widgets to applications. The second solution is to interpret the continuous data in real-time and have context widgets provide the discrete interpretations to applications.

All sensors have failure modes, making the data they produce unreliable at some point in time. To add to this problem, the data from many sensors must be interpreted to make sense of it. In much the same way as speech input must be recognized without a 100% accurate recognizer, context from sensors must also be understood. The problem with context is greater due to the fact that with speech input, the user is able to give feedback about incorrect recognition results. With context, this isn't the case. Recognized context is often used without being displayed to the user. The infrastructure must provide support for applications that may be using unreliable context information. This support many include the use of sensor fusion from multiple, heterogeneous sensors (with different failure modes) to increase reliability, as well as the passing of a reliability measure with each piece of context. We perform sensor fusion in an *ad hoc* fashion, but we are exploring a general mechanism for supporting it.

The infrastructure does not completely support the transparent acquisition of context for applications. In order for an application to use a widget, server, or interpreter, it must know both the hostname and port the component is being executed on. When the infrastructure supports a form of resource discovery [19], it will be able to effectively hide these details from the application. This will allow the application to really treat local context widgets like user interface widgets. When an application is started, it could specify the type of context information required and any relevant conditions to the resource discovery mechanism. The mechanism would be responsible for finding any applicable components and for providing the application with ways to access them. We have investigated many existing techniques for supporting resource discovery and are currently in the process of selecting one to implement.

Another limitation of the infrastructure is dealing with component failures. When a component fails, it has to be manually restarted. With a requirement for perpetual execution, this is clearly not an admirable property. The infrastructure does keep track of existing subscriptions between component restarts. So, when a component fails and is restarted, it knows what components were subscribed to it so it can continue notifying them of context events. But, the infrastructure needs a facility for automatically restarting components when they fail. We are currently investigating the use of watchdog processes and redundant components for this purpose.

6 Related Work

In our previous discussion on context handling, we discussed work that influenced us in our decision to treat context like input. We can see from the recent CoBuild workshop and AAAI Symposium on Intelligent Environments that there has been a lot of related work in smart environments. Rather than review the current state of smart environments, we will look at work that supports the use of context.

The proposed Situated Computing Service [10] has an infrastructure that is similar in intent to ours. It insulates applications from context sensors. It is a single server that is responsible for both context acquisition and abstraction. It provides both polling and notification mechanisms for accessing relevant information. It differs from our research in that it uses a single server to act as the interface to all the context available in an environment as opposed to our modular infrastructure. A single prototype server has been constructed as proof of concept, using a single sensor type.

The AROMA system [15] explored awareness in media spaces. Its architecture used sensor abstraction and interpretation to provide awareness of activity between remote sites. Interpreted context at each site was displayed at the other sites.

The CyberDesk system [6] used minimal context to provide relevant services to a desktop computer user. Its modular architecture separated context sensing, abstraction, and notification. It did not use a distributed architecture or support aggregation.

7 Conclusions

We have presented a context-based infrastructure for supporting the software design and execution of context-aware applications in the Aware Home, a prototype smart environment. Our infrastructure builds upon our previous work [17] that introduced the idea of a context widget for treating context like user input. We generated requirements for the infrastructure based on the differences between dealing with context and input. Using these requirements, we designed and built an infrastructure to make it easier for applications to deal with context. We discussed the benefits of the infrastructure through example applications that we have built. Finally, we described the limitations of the current infrastructure and, as part of our future work, plan to address these with the suggested improvements. We intend to test the context-based infrastructure through the use of a larger variety of context and the building of more complex applications within the Aware Home.

References

1. Arons, B. The design of audio servers and toolkits for supporting speech in the user interface. Journal of the American Voice I/O Society 9 (1991) 27-41.
2. Bauer, M. *et al.* A collaborative wearable system with remote sensing. In Proceedings of International Symposium on Wearable Computers (1998) 10–17.
3. Brotherton, J.A., Abowd, G.D. and Truong, K.N. Supporting capture and access interfaces for informal and opportunistic meetings. Georgia Tech Technical Report, GIT-GVU-99-06. (1998).
4. Clark, H.H. & Brennan, S.E. Grounding in communication. In L.B. Resnick, J. Levine, & S.D. Teasley (Eds.), Perspectives on socially shared cognition. Washington, DC. (1991).
5. Codella, C.F. *et al.* A toolkit for developing multi-user, distributed virtual environments. In Proceedings of Virtual Reality Annual International Symposium (1993) 401–407.
6. Dey, A.K., Abowd, G.D. and Wood, A. CyberDesk: A framework for providing self-integrating context-aware services. Knowledge-Based Systems 11 (1999) 3-13.
7. Dey, A.K. *et al.* The Conference Assistant: Combining context-awareness with wearable computing. To appear in the Proceedings of the International Symposium on Wearable Computers '99.
8. Harrison, B.L. *et al.* Squeeze me, hold me, tilt me! An exploration of manipulative user interfaces. In Proceedings of CHI'98 (1998) 17–24.
9. Horvitz, E. *et al.* The Lumiere Project: Bayesian user modeling for inferring the goals and needs of software users. In 14th Conference on Uncertainty in Artificial Intelligence (1998) 256-265.
10. Hull, R., Neaves, P., and Bedford-Roberts, J. Towards situated computing. In Proceedings of International Symposium on Wearable Computers (1997) 146–153.
11. Kidd, C, et al. The Aware Home: A living laboratory for ubiquitous computing research. To appear in the Proceedings of CoBuild'99 (1999).
12. Lamming, M. *et al.* The design of a human memory prosthesis. Computer 37, 3 (1994) 153-163.
13. MacIntyre, B. and Feiner, S. Language-level support for exploratory programming of distributed virtual environments. In Proceedings of UIST'96 (1996) 83–94.
14. Greenberg, S. Sharing views and interactions with single-user applications. In Proceedings of the ACM/IEEE Conference on Office Information Systems (1990) 227–237.
15. Pederson, E.R. and Sokoler, T. AROMA: Abstract representation of presence supporting mutual awareness. In Proceedings of CHI'97 (1997) 51–58.
16. Rekimoto, J. Tilting operations for small screen interfaces. In UIST'96 (1996) 167-168.
17. Salber, D., Dey, A.K., and Abowd, G.D. The Context Toolkit: Aiding the development of context-enabled applications. In Proceedings of CHI'99 (1999) 434-441.
18. Schilit, W.N. System architecture for context-aware mobile computing. Ph.D. Thesis, Columbia University (1995).
19. Schwartz, M.F. *et al.* A comparison of internet resource discovery approaches. Computing Systems (1992) 461-493.
20. Want, R. *et al.* The Active Badge location system. ACM TOIS 10, 1 (1992) 91–102.

ROBOTICS

Two Aspects of Multiple Mobile Robots: A Parallel Architecture and an Autonomous Navigation Strategy

S. Lauria, G.T. Foster, W. S. Harwin
Dept. of Cybernetics, The University of Reading, READING, U.K.
s.lauria@reading.ac.uk
http://www.cyber.rdg.ac.uk/research/

O. Habert
University of METZ, LASC, 57012 METZ cedex 1, FRANCE

Abstract

This paper describes a parallel architecture and a navigation strategy for multiple robots equipped with simple computational units and communication devices.

In particular, with the parallel paradigm, the aim is to consider such a collection of computational units as a cluster capable of performing parallel computations at a level similar to a classical massive parallel machine,albeit at a fraction of their cost. Furthermore, the autonomous navigation strategy discussed allows each robot to navigate in dynamic environments with little external guidance. Indeed, using data from its sensors the robot is able locate itself and detect possible changes in the environment. For this reason the Intelligent Building paradigm. on which the project is based, is presented. Moreover, the use of a particular form of neural network is presented to allow each single robot to detect changes in the environment. The initial experimental results from the MOBINET project presented here indicate that the technique is pratical and robust.

1 Introduction

This paper investigates a method for supporting navigation in large healthcare buildings based on work supported by the TMR MOBINET research network. Health care environments are generally well structured environments and with most of the activity inside the building it is advantageous to incorporate robot design into the building process. By allowing the building to provide some aspects such as navigation information or recognition tasks, the complexity and hence the cost of the robot can be lowered. Additionally networks already in the buildings can act as a management structure for multiple robots so that robotic services can be co-ordinated across the building. Structuring the environment in this way should not be perceived as a step backward in robotic research, but as the realisation that a more robust system is attainable through the utilisation of distributed computing resources present in the building.

Currently much of the research undertaken in the domain of mobile robotics is directed towards fully autonomous robots which tend to rely solely on the information gleaned about the environment from their own sensors. However, attempting to achieve a correspondence between observation and perception of physically present objects is a particularly difficult task, primary because any delay in actuation, as a consequence of prolonged sensor interpretation, can lead to undesirable behaviour. For an autonomous robot involved in, say, a delivery task from one location of the hospital to another, navigation information is required. There are serious problems with any autonomous robot attempting to generate a route plan in isolation. For any large building, an autonomous robot needs to store a global map of its environment. The robot will attempt to derive an optimum route to the desired location from its current position based only on what it has mapped from previous experience. Although the opinion of the robot is that the route is optimal, the current state of the environment may be different and the route far from optimal or feasible.

Although mobile robot systems currently installed in hospital environments can utilise global maps or global path generation from the building to plan a path to the target destination (see [1] for a detailed description), researchers are beginning to see the improvement that may be realised from the integration of intelligent environments into the design process for mobile robots, in having a general level guidance to the robot controlled by the intelligent distributed environment as it traverses the environment.

The second point analysed in this project aims to study a more local problem: the robot autonomous navigation. Indeed, as explained in more detail in section 4, the idea is to present a navigation model where each single robot, using information from its sensors, is able to navigate locally without any central guidance. The model is designed to meet the needs of handling real world complexities such as a dynamic environment, real time state change, incomplete information accessibility and distributed control. In this way the distributed system is dealing only with high level robot guidance and not with local navigation aspects related to each single robot (such as avoiding obstacles or incoming robot etc.). This choice allows to maximise the computational resources associated to the robot and use the building resources for more elaborated tasks.

2 Intelligent Buildings

As defined in [2, 3], an intelligent building is any infrastructure which supports the process of function of the building. The infrastructure in this paper utilises a distributed control network populated with intelligent nodes as illustrated in Figure 1. These nodes control various actuators based on the observation of the sensors under its control. All users are identified through the use of contact-less smart card. These contact-less smart cards are read by suitable microwave trasponders distributed throughout the building. Therefore any user or device associated with a smart card can be located to a particular region of the building.

The main components are now described.

Reader Node. This identifies the robot via a contact-less smart card as it enters its area of coverage and informs the Access Node. Readers Nodes are distributed to provide maximum coverage.

Access Node. The access node is connected to the wired computer network, which is part of the intelligent building infrastructure, allows each robot to access to the server resources as well as to other robots. Moreover, in this way all the computers can be clustered together as a parallel machine. In this way the intelligent building computational aspects are based on a parallel architecture.

Local Model. This model, as provided by the Access Node, gives the robot prior information about its immediate environment allowing the robot to correlate the data observed by its sensors [4]. Indeed, as explained later, the idea is to present a navigation model where each single robot, using information from its sensors, is able to locally navigate without any central guidance. The model is designed to meet the need of handling real world complexities such as dynamic environment, real time state change, incomplete information accessibility and a distributed control.

Network-based services. The network provides services in assisting the robot with its current task. For a robot involved in a transportation task, for example, the delivery of linen from the store to a nursing station, or assisting a patient to the day room, it calculates the optimum route and provides directional information as the robot travels the route using a parallel architecture [5, 6].

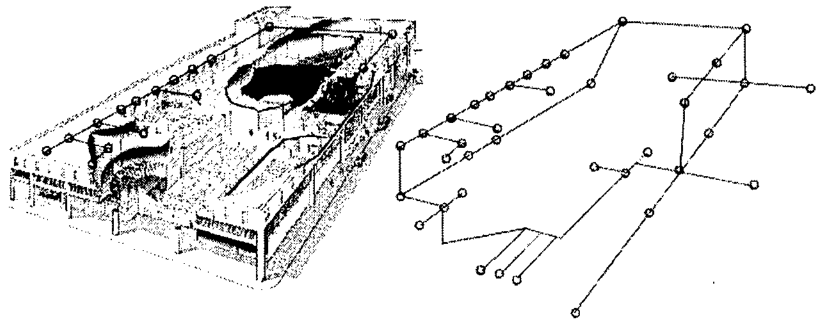


Figure 1: **The Intelligent Building.** *The building and the building representation using nodes as an abstraction for each room.*

3 The Parallel Paradigm

When dealing with a number of autonomous agents placed in a building and performing various tasks, there is the problem of wasted resources. Indeed, each robot is equipped with a simple computer and Network-based services are equipped with a set of computers. It is also desirable to use low cost resources (such as off- the shelves computers) as the number of agents can be very high. Moreover, these resources can be inactive for long time or for a well defined amount of time, as the robot can be inactive or performing tasks where there is no need for the computer. Nevertheless, the system (robots and the network-based services), or a part of it, can be involved in a task where high computational power is required (classify an object, find the shortest path between two points for a given robot, or even administrative duties if this network is part of an organization such as an hospital).

In this case, in order to reduce delays and problems, an optimization of the computing resources is desirable. The aim of this project is then to cluster together all these simple computational units. In other words, the idea is to take advantage of developing the agents/network-based service model as a Local Area Network (LAN). In this way, the network-based service has a distributed architecture where each robot computer is also part of the network-based service. Hence, all the resources, that a given instant, are not used for local purposes can, in this way, be involved with network-based service functions.

Recently, using LANs for such operations has already become an active research topic in similar areas such as industrial automation [7]. This approach offers several advantages. Indeed, once all these resources are wired together it is then possible to consider such a cluster as a Multi Parallel Processor (MPP). In this way, it is possible to perform parallel compositions (such as vision recognition) at a level similar to classical massive parallel machines but with no extra costs [8]. Indeed, only resources that are necessary, are used and a great advantage is obtained just by optimising the existing resources. More details, about this point are then given in [9].

A general criticism to parallel computing is based on the assumption that the needs of real computation will be met by new generations of uniprocessors with ever-increasing clock rates and ever-denser circuitry. However, parallelism is a cost effective way to compute because of the economics of processor development. The development costs of each new uniprocessor must be recouped in the first few years of its sales (before it is supplanted by an even faster processor). Once it ceases to be leading-edge, its price drops rapidly, much faster than linearly.

As parallel processing performance is less dependent than the uniprocessor one, on the processer performance, than this choice reduces the need of expensive resources in a parallel approach. For example if a generation i+1 uniprocessor is ten time faster than generation i processor, a multiprocessor built from generation i processors outperforms the new uniprocessor with only a hundred processors. Furthermore, the price of generation i processor is so low that multiprocessor is only small multiples of the price of the leading edge processor. Thus, in this project, the task is to optimize the existing resources and make effective some resource demanding computer tasks.

The novel approach used in this project deals with a further aspect. Traditionally the use of parallel programming requires specifically built multiprocessors, that is dedicated machines with very specific hardware and software architectures, which means still many difficulties on using them. By contrast, with the introduction of new recent technologies, an alternative approach has been used in this project. Indeed, the idea is to connect many off-the shelves computers (using common Operating Systems such as Linux or Windows95) to form a multi processor machine. In this way, it is possible to use such a cluster of computers spread all over the building for different purposes, such as robot object recognition, central strategy planning, administrative uses, at no extra cost.

This seems to be the ideal solution for situations, such as hospitals, where there are severe restrictions on financial resources and often with underused computer resources. Hence, this is the ideal solution for a situation such as an hospital with budget constraints. With such a choice, the system would be less dependent on the ultimate technology with great advantages from both an economical and efficiency point of view

4 Navigation Strategy

One of the most important and challenging aspects of map-based navigation is map-based positioning. Map-based positioning, also known as *map matching*, is a technique in which the robot uses its sensors to create a map of its local environment. This local map is then compared to a map previously stored in memory. As a results the robot can compute its actual position and orientation in the environment. The basic steps of a map matching algorithm are the following:

1. Acquire sensory information.
2. Build local map.
3. Establish correspondence between local map and stored map.
4. Calculate position.
5. Update stored map.

The main advantages of map matching are as follows [10].

- This method uses the naturally occurring structure of typical indoor environments to derive position information without modifying the environment.
- Map-based positioning can be used to generate an updated map of the environment. The new information can be used to update or define new features associated to that node in the topological model of the environment.
- Map-based positioning allows a robot to learn a new environment and to improve positioning accuracy through exploration.

Disadvantages of map matching are the specific requirements for satisfactory navigation. For example, map-based positioning requires that [10]:

- there be enough stationary, easily distinguishable features that can be used for matching. That is, a dynamic environment can seriously affect the final response.
- the sensor map be accurate enough to be useful.
- a significant amount of sensing and processing power be available.

The approach used in this paper aims to tackle some of the disadvantages listed above. In particular, with the use of a laser scanner it is possible to obtain very reliable data to build a map reducing also the amount of various sensors necessary. Moreover, as demonstrated in this paper, the algorithm presented here to build a map, integrating laser scans and odometer measurements, is robust and accurate and requires only few stationary features (such as walls and ceiling). Finally, some aspects of the algorithm are simplified in this project. For example, deciding which observations are new features previously not present in the map it is a crucial step to obtain a correct estimate of the position. With this approach such aspect has been bypassed. In the next sections the different steps of the map matching algorithm described above are presented.

4.1 Acquire Sensory information

In this project for the sensor-based spatial representation only a lidar (i.e. a laser scanner) as nowadays the measurement from laser scanners are extremely reliable and easy to process compared to other sources such as sonars, vision or infrared. The laser used is the SLK PLS-M5 rotating laser scanner [11]. Although the lidar scans a sector of 180, it provides 360 range measurements, one every 0.5 degree. For this reason and in order to have a complete 3-D data set, the laser has been mounted vertically rather than the standard horizontal position on a rotating platform. A motor controlling the rotation of the platform allows the robot to produce a complete 3-D set of data, where each vertical scan is providing a *slice* of the environment (from the ceiling to the floor). For the moment, it has been decided to produce 60 different vertical slices one every 6 degrees.

Each vertical slice is sent to an off the shelves computer installed in the robot controlling the synchronization between a vertical scan and the platform rotation. The disadvantage of such approach is the time required for obtaining a complete set of data. Indeed, although less than a second is necessary for each vertical scan, nevertheless the all process can take few minutes to be completed. A possible solution in order to reduce the acquisition time is to use a reduced rotation of the horizontal platform (That is a sector of 60 degrees rather than a complete 360 degree sector) as a default procedure, and only occasionally a complete rotation will be produced. This is justified with the assumption that during the robot navigation it is not necessary to continuously update areas of the environment not affecting the navigation (such as the area at the back of the robot, which the robot is leaving). Figures 2 (a) and (b) show respectively a typical 3-D set of data and a single vertical slice. Values in the X Y and Z-axes are given in centimetres and indicate the distance given by the laser scan.

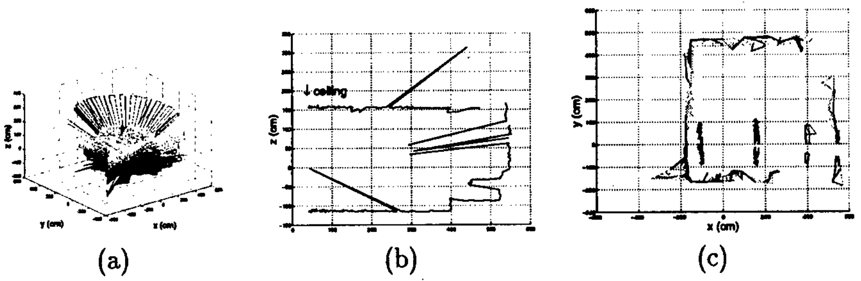


Figure 2: **The laser scans.** *Values in the X and Y-axes are given in centimetres. The robot is positioned at (0,0).* **(a)**.*A typical 3-D scan of the room.* **(b)**.*A single vertical scan. The horizontal lines are the floor and the ceiling. At 90 degrees there is the wall.* **(c)**.*View from the top of the same 3-D set of data showed in (a), but only considering a band of data from just under the ceiling. It is possible to distinguish the walls.*

The advantage with this approach is that in this way is possible to reduce time complexity tailoring the 3-D set of data for each specific task. For example, in order to allow a correct comparison between the stored map and the local map, it is important to handle data not affected by dynamical changes in the environment when the local map is built. For this reason, for each slice, only vertical wall distance measurements nearby the ceiling are considered (see Figure 2 (c)). In this figure robot is positioned in (0,0) and values are in centimeters. Whereas, when the detection of dynamical changes is necessary then only measurements away from the ceiling are considered. Nevertheless such approach is not immune from spurious data. Indeed, the presence of (static) irregularities, such as shelves nearby the ceiling, can bias the measurements. The method used in this project, and illustrated in the next section, aims to solve this type of inconvenient.

4.2 Build Local Map

4.2.1 Range Weight Hough Transform

The method used in this section is based on the extraction of walls using the Range Weight Hough Transform (RWHT) [12]. The Hough Transform was used on images for finding corners [13] and floor wall intersections. The range weighted version (RHWT) has then been introduced by [12] for 2-D range distance measurement rather than images. The originality of the project presented in this paper is on the use of 3-D data to handle spurious data and dynamical changes in the environment, without a significant increase of computational complexity. This section illustrates how, with the use of the Hough transform, it is possible to extracts flat surface elements from the lidar data. For a more accurate description see [14].

The range scan is a set of points in polar coordinates $\{r_i, \varphi_i\}$ relative to the distance r_i between the robot and an obstacle at the angle φ_i. The term *feature* refers to the element in the environment causing the flat surface in the range scan (e.g. walls, ceiling, etc.). Each feature is described by two parameters: the *orthogonal distance* d and the *direction* γ from the robot.

The g-weighted Hough transform $C(d, \gamma)$ is defined as:

$$C(d,\gamma) = \sum_i w(r_i \cos(\varphi_i - \gamma) - d) g(r_i, \varphi_i, \gamma) \tag{1}$$

where w is a window function and g is a weighting function to be selected. The argument in w is equal to the shortest distance between the point (r_i, φ_i) and the surface (d, γ). The function wx is defined as:

$$w(x) = \begin{cases} 1 & |\, x \,|\leq a \\ 0 & |\, x \,|> a \end{cases} \tag{2}$$

The simplest choice for the weight function g is the case $g \equiv 1$ making $C(d, \gamma)$ a histogram. The flat surfaces (e.g. walls, etc) can then be extracted as peaks. As also underlined in [12], in the range scan the sampling is uniform with angular steps δ_φ but not with distance steps. Indeed, as distances increase, each sample corresponds to a longer surface segment. For this reason, Range Weighted Hough Transform (RWHT) has taken in account this factor in the weighting function $g \equiv r_i$, giving:

$$C(d,\gamma) = \sum_i w(r_i \cos(\varphi_i - \gamma) - d) r_i \tag{3}$$

Each peak in the histogram gives an estimate of the perpendicular distance and the relative orientation of a surface. When searching for flat surfaces the reliability of the estimate can be increased using the information given by features associated to that node (i.e. that room). For example, for a rectangular room the estimate can be simplified by searching for groups of four peaks at 90 degrees intervals, rather than for single peaks.

4.2.2 Extracting the Local Map

This section describes how the RWHT is applied to the 3-D data to obtain the local map. In this paper the RWHT has been used twice with the set of data to simplify the RWHT computational complexity for a 3-D case.

First, RWHT has been applied to each single vertical scanner in order to extract vertical walls from it. Moreover, to further reduce the computational complexity, some information associated to the scanned node has been used. Each vertical scan is expected to have a flat surface for the ceiling at both known distance d_c and direction γ_c. Then, for each vertical scan, it is possible to detect the range measurement (r_c, φ_c) being the edge between the ceiling measurements $\{(r_1, 0), \ldots, (r_c, \varphi_c)\}$ and the remaining range scan. In this way, it is possible to select only a stripe $S \equiv \{r_{c+1}, \ldots, r_{c+k}\}$ nearby the ceiling where it is unlikely that any change has occurred since the last scan. Moreover, the vertical wall to be searched is known to be a certain fixed direction γ_w (usually 90 degrees (see Figure 2 (a))) with the robot. In this way, it is possible to apply the RWHT to S rather than to the entire vertical slice with the direction γ fixed to γ_w and varying just the distance parameter d. Figure 3 (a) shows an example of RWHT for a single vertical slice. The X-axis indicates the distance parameter (cm) used in the RHWT (corresponding to the Y-axis in Figure 3 (b)) and it ranges between 0 and 600 cm. The wall can be extracted as the distance with the highest peak in the RHWT histogram. Repeating this for all the m vertical lidar scans allows the system to obtain a very reliable set of distances $L \equiv \{r_1, \ldots, r_m\}$, where r_j is the wall distance calculated for the j^{th} scan. Figure 3 (b) shows the RWHT for the 60 different vertical scans. Figure 3 (c) shows the map obtained for the room shown in Figure 2, with the robot in position (0,0). In Figure 3 (c) the map obtained with the RWHT histogram (indicated with black dots) and the map from the laser scans from Figure 2 (c) (indicated with lines) are both showed to highlight their similarity. That is, with the use of RWHT it has been possible to obtain a reliable map (Figure 3 (c)) of the room (Figure 2 (c)) but at a fraction of the space storage.

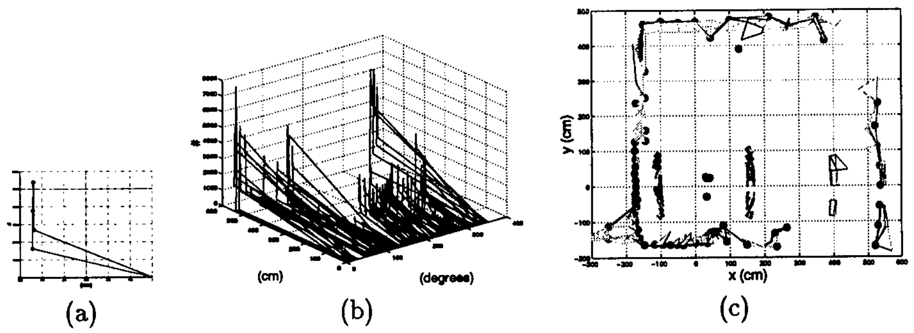
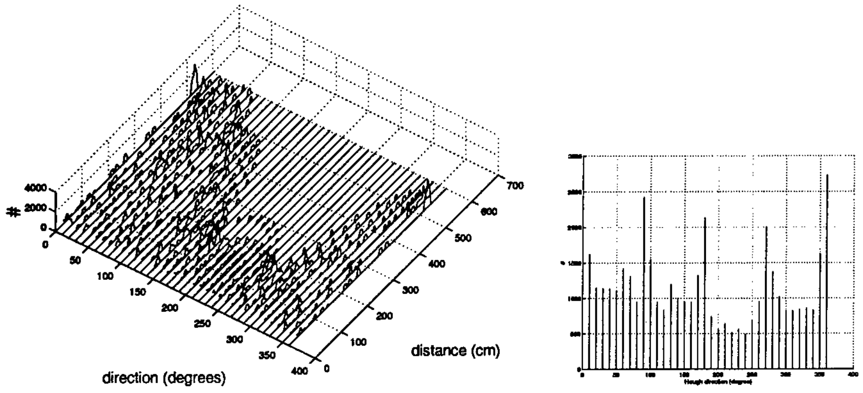


Figure 3: **Vertical RWHT.** **(a)**.*The RWHT for a single vertical scan.* **(b)**. *RWHT for 60 different vertical scans* **(c)**. *The map built using the values obtained from the RWHT is indicated with black dots. The map from 2 (c) is also showed with linesIn* **(b)** *the X-axis indicates the angles (degrees) and the Y-axis indicates the distance (cm).In* **(c)** *the X and Y-axis are in centimetres and the robot is in the position (0,0).*

Secondly, L can be the handled like the *usual* horizontal laser scan and the RWHT can be applied to it in order to extract the flat surfaces as showed in Figure 4. The advantage of using L is its robustness to spurious data and to dynamical changes with little computational complexity to be paid.

The walls are extracted by searching for the highest peaks in the RWHT (see Figure 4 (b). Hence, each wall is described by the perpendicular distance d_i and direction γ_i values associated to the i^{th}-peak. For example, in Figure 4 the four highest peaks correspond to the direction of the four walls (i.e. 90, 180, 270, 360 degrees) and the distances associated to the peaks match the distances of the walls. The vector $W \equiv d_1\gamma_1, \ldots, d_k\gamma_k$ can then be used to model the k-walls room. The local map so obtained can now be used for the next steps.



(a) (b)

Figure 4: **Horizontal RWHT. (a).** *The RWHT for the map from Figure 3.* **(b).** *The same histogram but in a 2D format (without the distances component). It is possible to note that the four highes peaks correspond to the direction of the four walls*

4.2.3 Local-Stored Map Correspondence and Position Extraction

Comparing the local map W with the stored map (obtained in a similar way but in a previous moment) it is possible to calculate the movement relative to latest checked position to obtain the new real position.

5 Conclusion

In this paper two aspects related to a multiple autonomous mobile robot scenario have been presented. It has been discussed the importance of approaching such aspects. At the same time, the proposed methodologies are also achievable as they have already been successfull in similar, altough not identical, areas.

The originality of this project is, then, in the attempt to unify these different methods as a whole and at the same time making them suitable for this case. Moreover, the use of simple equipments and the attempt to minimise the costs without a loss of performance render such a proposal very convenient, especially for governamental organisations (such as hospitals) with restricted financial resources.

The parallel implementation described in section 2 is the long time term aim. Before that, there is the short term priority to define and realise the autonomous navigation aspects. In particular, it is necessary to extract, from the data sensors, the necesary parameters and abstract representation, in the form described in section 4 reducing the relative time complexity.

6 Acknowledgments

Dr. Lauria and Dr. Hubert are pleased to acknowledge support for this work from the European Commission TMR project on MOBIle robot technology for healthcare services NETwork (MOBINET).

References

[1] O'Hart F., Foster G.T. "Creating Dynamic Robot environment from an intelligent building".

[2] G.T. Foster, D.E.N. Wenn, J.P.N. Glover. "ARIADNE - Exploiting A New Generation Of Intelligent Buildings." *Proc. 3rd TIDE Congress on Technology for Inclusive Design and Equality.* Helsinki, June 1998.

[3] Foster G.T. *Local Modelling Techniques for the Managment of Distributed Control Systems.* PhD Thesis. University of Reading, Reading.U.K. 1999.

[4] Foster, Glover, "Supporting Navigation Services within Intelligent Buildings."1997. *Proceedings of the 1st MobiNet Symposium on Mobile Robotics Technology for Health Care Services. pp 261-270.*

[5] Harwin W., Foster G., Hu H. Stirchombe D. "An Agent Based Approach to Delivery Services in a Hospital Environment." *2nd Mobinet Symposium.* Edinburgh, July 1999.

[6] Foster G.T., Wenn D., O'Hart F., Harwin W., Generating Virtual Environments To Allow Increased Access to The Built Environment. In *International Journal of Virtual Reality, No. 4,* 1999.

[7] Jiang, "Wireless Communications and Priority Access Protocol for Multiple Mobile Terminals in Factory Automation". *IEEE Trans on Robot and Automation,* Vol. 14, No 1, February 1998. pp137-143.

[8] Lauria M., Chien A. "MPI-FM: High Performance MPI on Workstation Clusters." *Journal of Parallel and Distributed Computing, Vol. 40, No. 1, pp. 4-18, January 1997.*

[9] Lauria S., Patel R., "Multiple Mobile Robots: A New Parallel Architecture".1999. Internal Report.

[10] Everett, H. R., *Sensors for Mobile Robots. Theory and Application.* A K Peters Ltd. 1995.

[11] PLS-M5. SICK optic electronic.

[12] Forsberg J., Larsson U., Wernersson A., "Mobile Robot Navigation using the Range-Weighted Hough transform". *IEEE Robotics and Automation Magazine.* Vol 2, n. 1 pp. 18-26, March 95

[13] Kosaka A., Kak A., "Fast Vision-Guided Mobile Robot Navigation Using Model-Based Reasoning and Pradiction of Uncertainties", *Image Understanding,* Nov. 1992.

[14] Larsson U., Forsberg J., Wernersson A., "Mobile Robot Localization: Integrating Measurements from a Time-of-Flight Laser". *IEEE Transactions on Industrial Electronics,* vol43, n.3, pp. 422-431. June 1996.

REALITY & VIRTUAL REALITY IN MOBILE ROBOTICS

B.R. Duffy, G.M.P. O'Hare, R.P.S. O'Donoghue, C.F.B. Rooney, R.W Collier

PRISM Laboratory, Dept. of Computer Science, University College Dublin (UCD), Belfield, Dublin 4, Ireland
{Brian.Duffy, Gregory.OHare, Ruadhan.ODonoghue, Colm.Rooney, Rem.Collier}@ucd.ie

Abstract. This paper advocates the application of VRML in the visualisation of social robotic behaviour. We present the Virtual Reality Workbench, which via the Social Robot Architecture integrates the key elements of Virtual Reality and robotics in a coherent and systematic manner. To support this, we deliver a development environment, Agent Factory, which facilitates rapid prototyping and visualisation of social robot communities.

Introduction

The use of Virtual Reality has developed from the traditional game playing metaphor to application as diverse as e-commerce and virtual communities. This paper proposes the use of VRML in the visualisation of intelligent agent communities. We present the Virtual Robotic Workbench, which by monitoring the mental states of agents in a multi-agent system, can display and update a VRML representation of the agents environment. An obvious application of such technology is robotics.

The *Social Robot Architecture* (SRA) combines reactivity, deliberation and social ability to enable robots to deal competently with complex, dynamic environments. We demonstrate how the Virtual Robotic Workbench can present a virtual window into the robots' environment, allowing for remote experimentation and thus providing insights into the inconsistencies between a robot's mental image of an environment and the physical counterpart. Figure 1, below, presents an architecture, which seamlessly integrates, real world robots, multi-agent development tools, and VRML visualisation tools into a coherent whole.

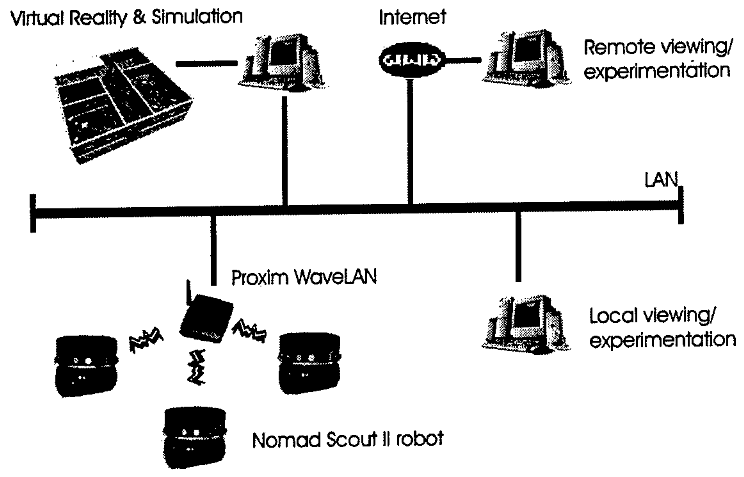

Virtual Reality & Simulation
Internet
Remote viewing/ experimentation
LAN
Proxim WaveLAN
Local viewing/ experimentation
Nomad Scout II robot

Fig. 1. Social Robot: The Coherent Whole

Virtual Reality

VRML (Virtual Reality Modelling Language) is a recent advancement in Internet technologies [TSW+96]. VRML allows for the development of dynamic 3D worlds, which can be viewed through a web browser. VRML 2.0 provides the user with many tools facilitating scene animation and scene update based on user input. Some applications of the technology are as follows:

Virtual communities

A substantial area of VRML research is the provision of software tools which enable user immersion in a *connected virtual community*. Several such VR toolkits are available which enable the creation and deployment of multi-user interactive virtual communities. One such tool is VNET [VNET], which is a VRML and Java based 3D Virtual World client/server system, freely available under the GNU Public Licence.

E-commerce

ViSA (Virtual Shopping Agent Architecture) [OO99] is representative of a new generation of e-commerce system that enables the user to truly enter the retail arena and participate in the virtual shopping experience. The ViSA System offers: immersion within a 3-D shopping environment and virtual community of shoppers; intelligent assistance in all stages of product procurement; contextualised and personalised shopping experience for individual shoppers.

Robotics

Many exemplary systems demonstrate the use of VR for simulation and visualisation within robotics. Here we consider but four subsystems.

Workspace® [Workspace] from Robot Simulations Ltd. is one such tool. It allows the user to model and display simulations of workcell layouts involving conveyors, AGV's and the evaluation of their performance. System programming can take place before robot installation or while an existing workcell is in operation. Complex layouts can be tested in complete safety either in an office environment or on the shop floor. Simulation highlights potential problems before they happen, allowing quick and easy modifications.

The RHINO-project [BCF+98] a joint project between the Institut für Informatik III of the Rheinische Friedrich-Wilhelms-Universität Bonn and the Carnegie Mellon University (USA). The central scientific goal of the RHINO project is the analysis and synthesis of computer software that can learn from experience. The team believes an essential aspect of future computer software will be the ability to flexibly adapt to changes, without human intervention. The ability to learn could soon enable robots like RHINO to perform complex tasks, such as transportation and delivery, tours through buildings, cleaning, inspection, and maintenance.
While giving tours in the Deutsches Museum, RHINO can be observed and even tele-operated through the Internet via a virtual reality medium. RHINO will make available on-line camera images recorded in the museum.

The DLR VRML 2.0 Robot [Roh97] is a robot simulation system that enables the telemanipulation of real robots via WWW. It provides a detailed graphical model of the robot that can be manipulated intuitively using the mouse. Basic features are forward and backward kinematics with various interaction possibilities for motion in joint space or Cartesian space. Routes can be programmed and edited. The software also handles the connection and control of various tools.
It is anticipated the VRML interface will support the operation one of the KUKA robots.

RESOLV Project [LGL+98] provides a working model of its Autonomous Environmental Sensor for Telepresence (AEST). This robot tours the inside of a building and automatically creates a 3-D map of the interior compete with surface texture information. The 3-D reconstruction module produces two models; a geometrical model suitable for conventional CAD systems, and another composed of triangular meshes suited to graphical visualisation. Both representation use VRML format and are thus viewable with any WWW browser.

In the next sections we present the Social Robot Architecture and the use of Virtual Reality for the visualisation of social robot communities as opposed to individual robots.

The Social Robot Architecture

The *Social Robot* Architecture aims at achieving team building and collaborative behaviour through the judicious synthesis of the reactive model with that of the deliberative model. The architecture (figure 2) is comprised of four discrete layers: physical, reactive, deliberative developed using Agent Factory, and social.

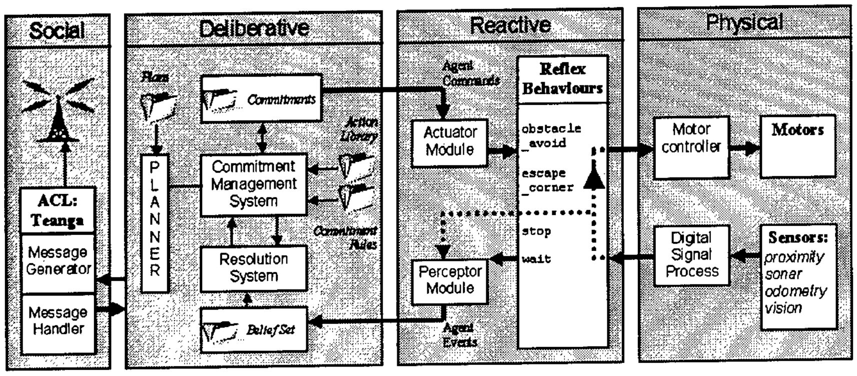


Fig. 2. The Social Robot architecture: The Robot Agent

Physical: Robots in terms of this research may take the form of either that of a physical entity, specifically the Nomad Scout II or a simulated entity (Nomadic Technologies XRDev robot).
Reactive: A series of fundamental reflex behaviours are implemented at this level. The sensory information is processed resulting in clear *agent_events* and communicated to the deliberative level. *Agent_commands* are received from the deliberative layer.
Deliberative: This comprises of a Belief Desire Intention (BDI) [RG91] architecture developed through Agent Factory. The perception process deals with converting agent events into beliefs and adding them to the belief set providing the agent with an up to date model of its current perceived situation and results in the agents' commitments being updated accordingly. Pre-existing commitments are analysed and those pertaining to the current time frame honoured resulting in either a communicative act being sent to the social level or a physical act being passed to the actuators via the reactive level.
Social: Our agents interact via an Agent Communication Language (ACL), entitled Teanga.

More detailed information on *The Social Robot Architecture* and its applications are given in [DCO+99].

Agent Factory

Agent Factory has been developed to facilitate the rapid prototyping of Multi-Agent Systems. The system offers an integrated toolset that supports the developer in the instantiation of generic *agent structures* that are subsequently utilised by a pre-packaged agent interpreter that delivers the BDI machinery. Other system tools support interface customisation and agent community visualisation.
In creating an agent community three system components must be interwoven, those of *agents*, a *world* and a *scheduler*.
The *agent* is the core computational unit underpining Agent Factory, it combines a series of attributes that represent and support the Mental State model of an agent, a set of methods (the actuators), a set of perceptors, an Agent Communication Language (ACL), and a Commitment Revision Strategy. This design is then executed using a generic *Agent Interpreter*. The *scheduler* controls execution of the community, using an algorithm that exploits parallelism where possible. Finally, the *world interface* acts as a medium between the problem domain, the community it is being developed for, and the other components of the Agent Factory System.
The creation of an agent community is facilitated by the *Agent Factory Development Environment*, which provides a *Component Library* and a selection of tools for the rapid prototyping of agent communities. The Component Library is built from Component Design Hierarchies (CDH) that extend the standard Object Hierarchies in the OOP Paradigm.
The *Agent Factory Run-Time Environment* provides the support necessary for the release of a completed Multi-Agent System. This environment comprises of a *Run-time Server* and an *Agent Interpreter*. The Run-time Server offers two main services: access to non-agent components of the system (a controller & some worlds), and a set of tools for viewing and interacting with the agent community. The Agent Interpreter provides the mechanisms by which the agents may be executed and visualised. Access to these environments is provided both locally through Graphical User Interfaces (GUIs) and remotely through the World Wide Web (WWW) via a purpose built *Web Server*.
The Agent Factory System has been discussed more completely elsewhere in the literature [CO99].

The Virtual Robotic Workbench

One of the key tenants of our research has been the provision of multiple views of multiple robot systems. The primary view is the physical perspective of the Nomad Scout II's navigating the physical world. The secondary, more abstract view, is a virtual reality perspective provided via the Virtual Robotic Workbench, which delivers a 3-D VRML world via the Internet (figure 3).

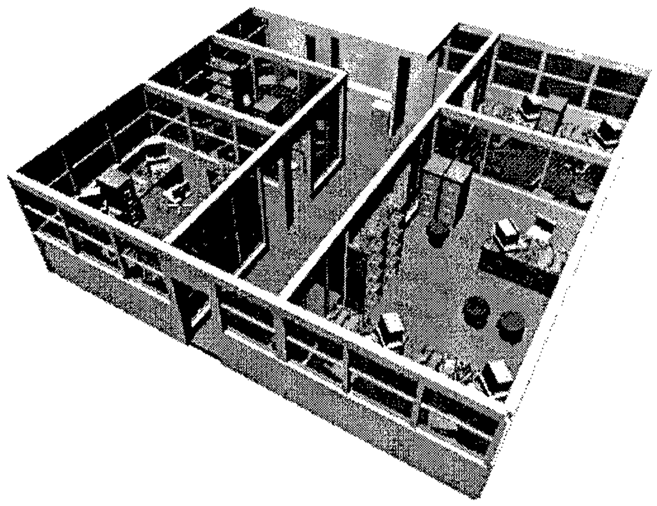


Fig. 3. Virtual reality view of the IMPACT research environment

Herein we harness the advantages of using virtual environments, by directly relating virtuality and reality in a seamless manner. This permits multiple views, information hiding and abstraction, system interaction, and behaviour scrutiny via snapshots and recordings. A *Virtual Robotic Workbench* provides a medium through which researchers can design robot experiments remotely and without recourse to the purchase of expensive robotic entities in the first instance. Researchers can articulate their experiments across a Java interface whereby they tune certain key workbench parameters, namely:

- The Customisation of the Environment
  - The Number of Robots;
  - The world within which they are to be situated;
- The Customisation of the Robots
  - Their Name;
  - The visualisation avatar associated with each robot;
  - Mapping virtual robots to physical robots;
  - The behaviours ascribed to a given robot;
  - Their Initial Location within the selected world;
- The Task
  - The selection of the shared goal;

Figure 4 depicts the Virtual Robotic Workbench interface by which these parameters are specified. Once the experiment has been crafted the subsequent activation results in the observable behaviour of the robots across any suitably configured web browser. The behaviour of the robots is governed by the Social Robot Architecture with the higher level functions directly furnished through Agent Factory. As such, the navigation and behaviour of the robotic avatars can be seen by utilising the Agent

Factory Visualiser. Detailed treatment of the Agent Factory Visualiser is presented elsewhere in the literature [OCC+98].

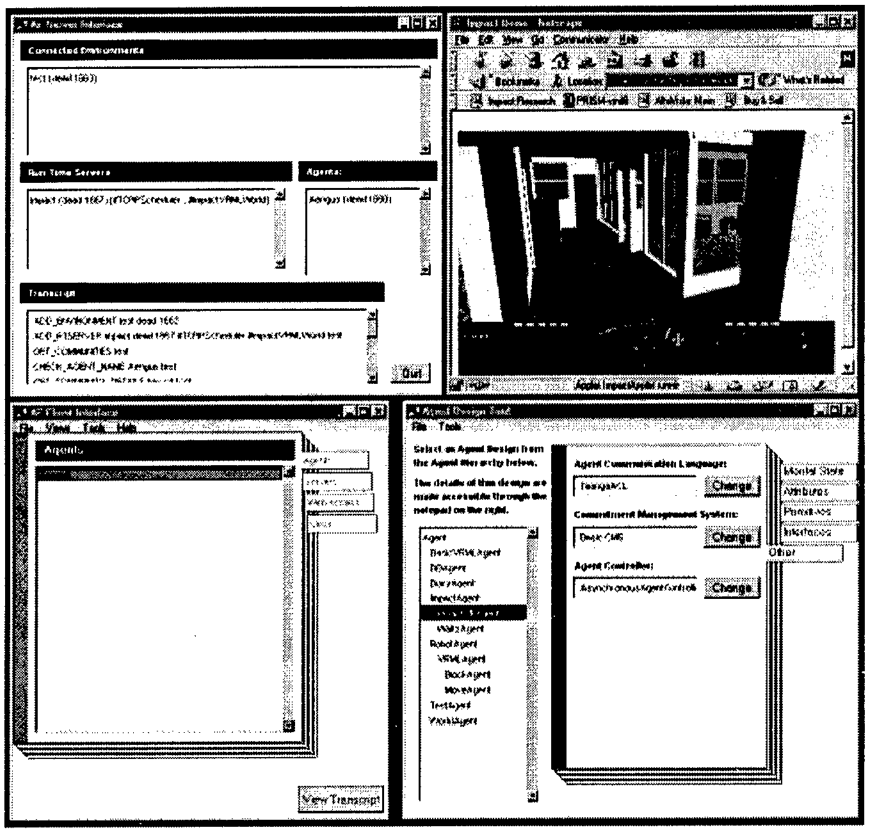


Fig. 4. The Virtual Robotic Workbench

The benefits of the application of virtual reality to robotic experimentation are manyfold. Perhaps, the most obvious is that it allows for remote behaviour observation from multiple perspectives.

The Virtual Reality Visualiser

Within the Social Robot Architecture, the Run-Time Server performs the particular role of housing the machinery for the delivery of the *Virtual Robotics Workbench.* This workbench facilitates the remote specification and subsequent observation of Social Robotic experiments. Within the context of Agent Factory, the Virtual Robotic Workbench utilises an existing tool, the Agent Factory Visualiser (AFV) to present a 3D view of the agents. An earlier version of this tool is discussed in [OCC+98]. The

Visualiser tool facilitates the presentation of Virtual Reality views of the agent community commissioning standard web browsing technologies. It is comprised of three components:

- The AFV Server: This component is plugged into a Run-Time Server and acts as a proxy server between the agents and the AFV Clients. The server maintains lists of objects depicted in the Virtual Reality Scene.
- The AFV Client: This is a Java Applet embedded into a HTML page. The applet is responsible for the maintenance of a current 3D view of the agent community. This page is viewed within a standard Web browser that has support for the VRML.
- A Web Server: The third component is a standard web server. This is required to host the AFV Client in order that it may be downloaded as necessary.

We now discuss the interplay between these components. The key component of this system is the AFV Client, which is responsible for the delivery of the VR view of the agent community. As indicated earlier, the AFV Client is a combination of standard web technologies. A client interface is comprised of a VRML Scene which is updated through a Java Applet. Upon loading the AFV Client, it is the responsibility of the user to connect the client to a suitable AFV Server. After the AFV Client and Server are connected, the next step is for the Server to send the URL of the VRML scene to be loaded (i.e. the agents arena).
The update of the VRML Scene is achieved through an interface provided as standard within the VRML97 specification. This interface is called the External Authoring Interface (EAI). The purpose of the EAI is to enable the modification of VRML scenes through a pre-defined library of functions. These functions are made available through a collection of Java classes that are used within the AFV Client. Available functionality includes the ability to create additional VRML objects within the VRML scene, and to modify parts of the VRML scene.
Henceforth, those agents that are to be displayed within the VRML scene are appropriately configured, and subsequently appear. This configuration necessitates permits the subsequent *tapping* of the *agent_event* queue as described later. From this point on, the agents' movements are mirrored in the VRML Scene.

Figure 5 illustrates how an agent's virtual position may be updated based upon its physical position. *Agent_events* are triggered within the reactive layer of the SRA and transmitted to the deliberative level. These events relate to the detection of landmarks such as corners of a room, doorways etc. For example, the detection of a landmark results in system messages such as the following being sent: '*AGENT-EVENT LANDMARK corner AT 5.32 4.14*'. This states that a corner was detected at position (5.32, 4.14) in the agent's local coordinate system.

Each agent handles *agent_events* about that event. However, the update of the Virtual Robots' position does not occur through any deliberative action on the part of the agent. Instead, a *tap* is placed upon the event queue, which listens for a landmark *agent_event*. Upon detection of this event, the coordinates are taken and converted to the absolute coordinate system used by the Virtual Agents. Subsequently a system message informs the Visualiser to update the position of a given virtual robot. Such a message takes the form of: '`MOVE_OBJECT <name> <x> <y> <orientation>`' informing the Visualiser to move the object with the given name to position `(<x>, <y>)`, and to rotate it to the given orientation. The AFV Server receives this message and propagates it to all the AFV Clients currently connected. These clients receive the message and update their VRML scenes accordingly.

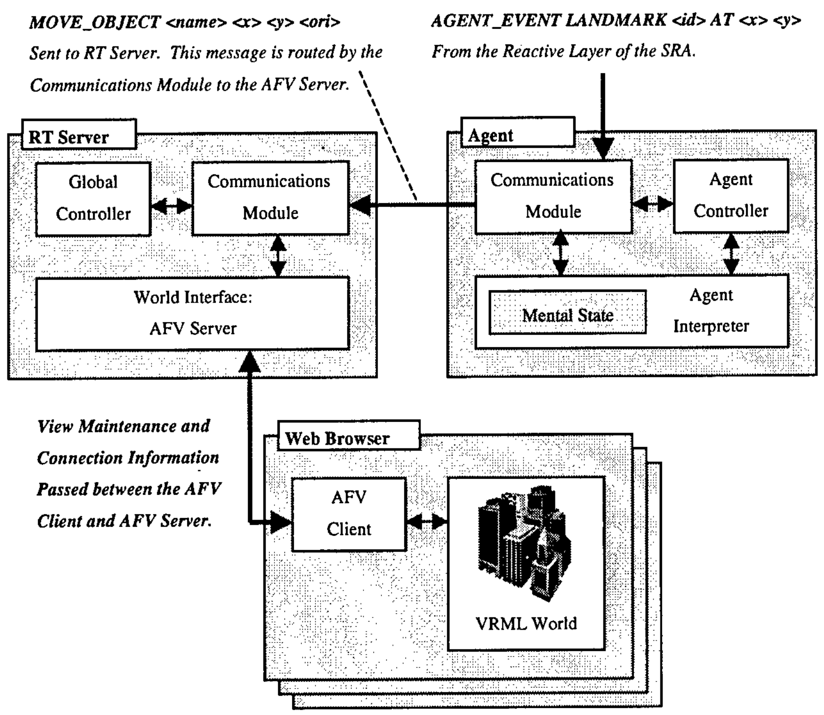

MOVE_OBJECT <name> <x> <y> <ori>
Sent to RT Server. This message is routed by the Communications Module to the AFV Server.
AGENT_EVENT LANDMARK <id> AT <x> <y>
From the Reactive Layer of the SRA.
RT Server
Global Controller
Communications Module
World Interface: AFV Server
Agent
Communications Module
Agent Controller
Mental State
Agent Interpreter
View Maintenance and Connection Information Passed between the AFV Client and AFV Server.
Web Browser
AFV Client
VRML World

Fig. 5. Position updating within the VR Workbench

This results in a real-time link between the real robot and its virtual counterpart. In the next section we examine how to maintain a sufficient degree of positional accuracy to make this link viable.

Experimentation

Robot experiments are characterised by firstly selecting a world, subsequently situating robot(s) in this world, and finally ascribing behaviours to these robots. In this approach the odometric information sent by the real-world robot to Agent Factory is supplemented with sonar and visual information that may indicate detected environmental features, and relative distances from these features. Such sensor fusion enables the robot avatar position update not only by mirroring the uncertain real-world coordinate position updates, but also by matching current sensory information with the VRML world to reduce the uncertainty of the robot's position.

Of course, we do not aspire to have a completely accurate real-time synchronisation between real and virtual robots. The VRML view is primarily a visualization tool and therefore it is sufficient to update and recalibrate the virtual world stochastically.

The problems associated with obtaining robot position accurately without the use of tailored environments (by providing landmarks etc.) are rife [LMK+97] [Saf96] [DN99] [SC93] [RL97] [HR96]. Robot research has extensively documented problems associated with the cumulative positional error in the robots' own odometric sensor readings render such sensors inadequate as exclusive sources of long-term positional information. Given the robots' internal position information (x_r,y_r) the problem is to ensure that the robot avatar is in *approximately* the corresponding location (x_v,y_v) in the virtual world (see figure 8).

The first step in achieving this synchronization task is to send the actual robot position update to Agent Factory. This event takes the form of `position_update Agent(`x_1, y_1`)`. This coordinate is based on the robots internal odometry system and may therefore harbour inaccuracies. Based on this information the robot avatar position is provisionally updated. However, to avoid the certain inconsistencies that would arise between the two worlds, we employ several recalibration techniques, which operate on both a local and a global basis.

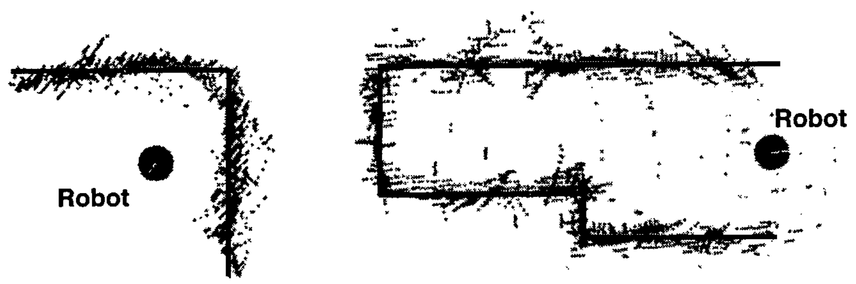
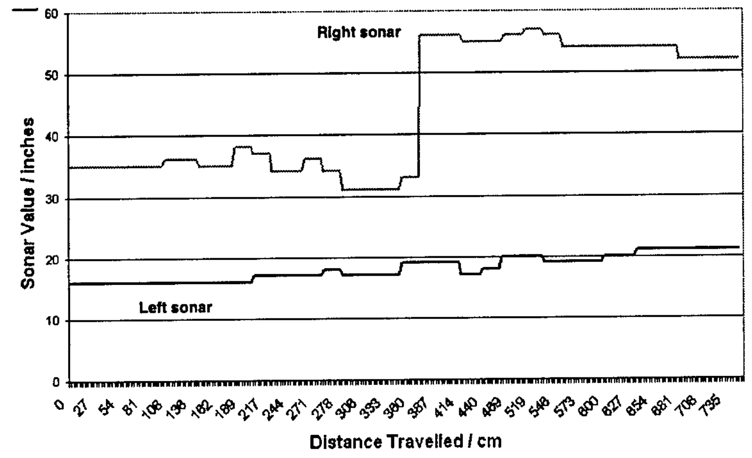


Fig. 6. (a) Corner detection, (b) Wall detections

The world is comprised of predefined locations. On passage between these locations, an event is sent to Agent Factory. For instance, as the real robot passes through a door (recognized by both visual analysis and by its characteristic sonar signature) an event is sent such as *door(bui,sonar_number,distance)* which corresponds to Bui's belief that there exists a door at distance *distance* from sonar *sonar_number*. On the occurance of such an event, we may update the avatar position accordingly, by zeroing both the real and the virtual position coordinates. Global recalibration is thus achieved.

On entering a new location, we can retrieve the robot's position with a certain degree of accuracy. We may carry out local recalibration based on the various features which may be detected within in a specific location. Common features that can be recognised using sonar or vision (or both) include doors, walls, primary corners, desks etc. Figure 6 shows corner and wall detection from sonar data. These features may be

classed as primary location cues towards achieving local recalibration for accurate visualization through the VR medium. This results in a globally relevant perspective of the environment through VR. Having recognized such a feature, and given that we have an estimate of true robot position, we may update the position estimate by *fitting* the detected feature to the corresponding feature in the VRML model.
Here we merely present initial feature extraction results.



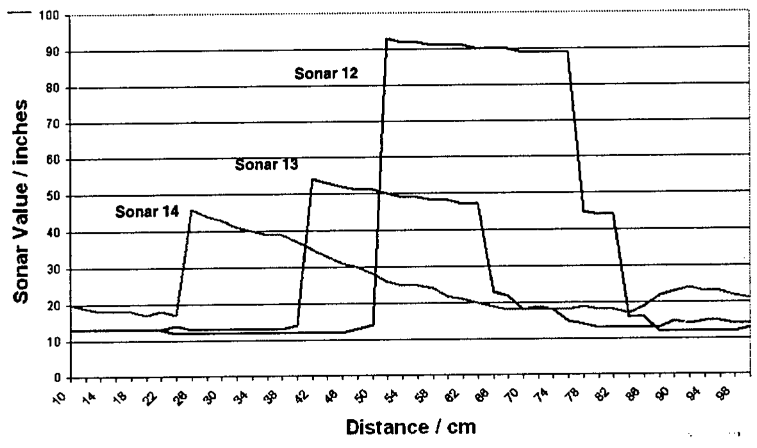


Fig. 7. Sonar signatures: (a) corridor; (b) door

Figure 7(a) graphs the sonar readings from the left and right sides of the robot as it travels parallel with the corridor walls. Figure 8(a) depicts the (real-world) corridor in which a robot experiments was performed, while figure 8(b) depicts the VRML representation of this corridor. The relationship between the graph in figure 7(a), and the views of the corridor should be obvious. The rather large jump in the graph of the *right sonar* corresponds to the increase in distance from the right wall as the robot passes the filing cabinet on *its* right hand side, whereas the flat plot of the *left sonar* corresponds to the straight wall the robot was travelling parallel with. It is relatively easy to fit a line to the plot of the left sonar, and extract the event `wall_from(`x_1`,` y_1`,` x_2`,` y_2`, sonar_number, distance)`, where x_1,y_1 and x_2,y_2 represent the perceived start and end points of the wall, which is distance *distance* from sonar *sonar_number*. In most cases the task of matching this to the VRML representation of this wall is feasible, and having achieved this the robot position and orientation can be fine-tuned.

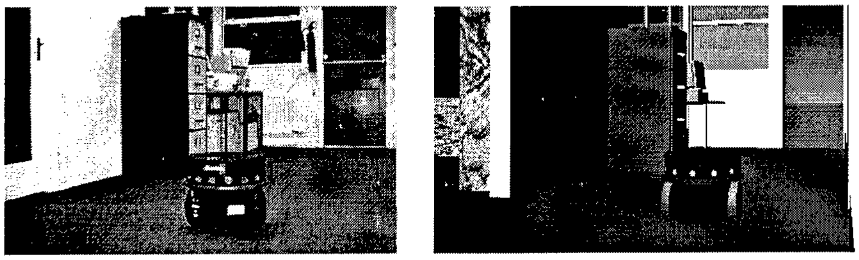


Fig. 8. The virtual reality and physical reality in parallel

Figure 7(b) is a graph of the readings from three of the robot's sonar as it passes an open door. This sonar footprint is characteristic of an open door. First sonar 14[1] jumps as the free-space in the new room is detected. Next sonar 13 makes a similar jump, and finally sonar 12, the side sonar jumps up too, indicating that the door is directly to the side of the robot. While the value from sonar 12 jumps, the values from sonar 13 and sonar 14 fall off again. This *sonar signature* reappears each time a door is passed. It differs from a corner sonar typical corner signature in that sonar 13 and sonar 14 decrease again in the door case. In addition to recognising the door through its sonar signature, we have successfully employed visual analysis for door recognition to provide corroborative evidence of the presence of a door. A library of sonar signatures can therefore be utilised for feature recognition.

Stochastic Synchronisation Between Worlds

The trade off between perception and representation has been extensively documented within AI literature. In a conventional sense mobile robots sense (perceive) their environment as they navigate through it. The emphasis is clearly upon perception rather than representation. Within the Social Robot Architecture we redefine this trade off. Perceptions are transformed into beliefs about the environment within which the robot exists. As such, perceptions underpin a subsequent thin representation of the environment. Individual robot perceptions and corresponding beliefs may subsequently be propagated to fellow robots in keeping with the social nature of our architecture. Thus, given the inherently myopic nature of our Nomads,

[1] Sonar 12 is directly on the side perpendicular to the wall with sonars 13 22.5° and 14 45° to the front respectively

an inexact and incomplete representation of the environment is derived through the augmentation of other partial views communicated by fellow robots. Currently, Agent Factory agents are gullible and as such beliefs are received in good faith and directly incorporated into the mental state.
When we contrast this with our virtual robots then the perception representation trade off is somewhat different. Our virtual robots have currently no perception capability. They do however have a fairly rich representation of the virtual environment, a direct replica of its real counterpart. By taking receipt of Agent Factory events the world is refreshed to reflect robot transformations. A vast amount of sensory data is culled by the Nomads and filtered through key feature footprints, which encode patterns of sonar data synonymous with particular features. Upon recognition the associated event is generated, dispatched to Agent Factory, and the next cycle the belief update effected. Subsequent to this the Virtual world is updated. As such a synchronisation of sorts is achieved. Rather than this being continuous it is stochastic in nature. There are a multitude of problems associated with achieving the former. For our purposes, the approximate depiction of robot behaviour in virtual robot experiments viewed and expressed through the medium of the Internet, the latter proves adequate.
To date we have concentrated upon the flow of information from the real world to the virtual. A counter flow of data could be harnessed and as yet we have not truly investigated this. Given the richer representational model held by the virtual environment (i.e. a building comprised of floors, in turn comprised of rooms interconnected by corridors and populated with objects and robots) upon recalibration of the robot position, beliefs about the immediate location could be funnelled back to the robot agents. For example the existence of a door to the immediate left. This bi-directional information flow seems to offer mutual benefit.

Discussion and Conclusions

Within this paper we have characterised our research on Social Robotics and specifically the interplay between virtual and real environments for Social Robot experimentation. The creation of the Virtual Robotic Workbench offers a medium through which cost effective robot experiments may be conducted. In particular we access robot behaviours in a variety of virtual buildings.

Within the context of this work we regard robots as active intelligent objects. We would envisage that smart buildings would be comprised of a collection of intelligent interacting components some of which might be static (Heating Subsystem, Lighting Subsystem, Entry Controller) or dynamic (robots, AGVs, Lifts). Robot agents in the context of the Social Robot Architecture permit effective collaboration through their social nature. We advocate an agent community for the delivery of the intelligence necessitated in such smart buildings. Thus collaboration between a lift agent and a robot agent and entry control agent could prove crucial for floor migration. Collaboration between the lighting control agent and a robot agent could facilitate lighting adjustment for onboard frame grabbing facilities for visual perception.

The application of social robotics within the context of smart environments and buildings is endless. Two possible scenarios are, the self-cleaning building and mail delivery robot teams. At appropriate times a team of social cleaning robots could collaboratively effect an exhaustive vacuuming activity. Alternately a team of mail delivery robots could plan delivery schedules for internal mail dispatch.

Acknowledgements

We gratefully acknowledge the support of Enterprise Ireland through grant No.'s SC/96/621 COMEDY and SP/97/08 IMPACT (Initiative for Mobile Programmable Agent-based Computing Technologies).

References

[BCF+98] Burgard, W., Cremers, A.B., Fox, D., Hähnel, D., Lakemeyer, G., Schulz, D., Steiner, W., and Thrun, S. "The Interactive Museum Tour-Guide Robot", Proceedings of the 15th National Conference on Artificial Intelligence (AAAI'98), Madison, Wisconsin, 1998

[CO99] Collier, R.W., O'Hare, G.M.P., "Agent Factory: A Revised Agent Prototyping Environment", 10th Irish Conference on Artificial Intelligence & Cognitive Science, 1-3 Sept., 1999 University College Cork, Ireland

[Col96] Collier, R. "The realisation of Agent Factory: An environment for the rapid prototyping of intelligent agents", M.Phil., Univ. of Manchester, 1996

[DCO+99] Duffy, B.R., Collier, R.W., O'Hare, G. M. P., Rooney, C.F.B., O'Donoghue, R.P.S. "SOCIAL ROBOTICS: Reality and Virtuality in Agent-Based Robotics", in Proceedings of Bar-Ilan Symposium on the Foundations of Artificial Intelligence: Bridging Theory and Practice (BISFAI), 1999.

[DN99] Duckett, T., Nehmzow, U., "Self-localisation and autonomous navigation by a mobile robot", Proc. Towards Intelligent Mobile Robots 99 (Bristol). Tech. Report Series, Dept. Computer Science, Manchester University, Rep. No. UMCS-99-3-1. 1999.

[HR96] Harris, K.D. and Recce, M. (1996) Localisation and goal-directed behaviour for a mobile robot using place cells in Proc. Sintra Workshop on Spatiotemporal Models in Biological and Artificial Systems, IOS Press, Amesterdam.

[LGL+98] Leevers, D., Gil, P., Lopes, F. M., Pereira, J., Castro, J., Gomes-Mota, J., Ribeiro, M. I., Gonçalves, J. G. M., Sequeira, V., Wolfart, E., Dupourque , V., Santos, V., Butterfield, S., Hogg, D., Ng, Kia. "An Autonomous Sensor for 3D Reconstruction" 3rd European Conference on Multimedia Applications, Services and Techniques Berlin-Germany, 26-28 May 1998 ECMAST'98

[LMK+97] Lambrinos, D., Maris, M., Kobayashi, H., Labhart, T., Pfeifer, R., and Wehner, R. (1997). "An autonomous agent navigating with a polarized light compass", Adaptive Behavior, 6, 131- 161.

[OCC+98] O'Hare, G.M.P., Collier, R., Conlon, J. and Abbas, S., "Agent Factory: An Environment for Constructing and Visualising Agent Communities", Proc. AICS98, 1998

[OJ96] O'Hare, G.M.P., Jennings, N.R., (Editors.), Foundations of Distributed Artificial Intelligence, Wiley Interscience Pub., New York, 1996, 296 pages. ISBN 0-471-00675

[OO99] O' Sullivan, G., O' Hare, G.M.P., Coughlan, C., 1999. "ViSA: Virtual Shopping Agent Architecture" Submitted to 8th Euromicro Workshop on Parallel and Distributed Processing Rhodos, Greece, January 2000.

[RG91] Rao, A.S., Georgeff, M.P., "Modelling Rational Agents within a BDI Architecture", Prin. Of Knowl. Rep. & Reas., San Mateo, CA., 1991

[Roh97] Rohrmeier, M. "Telemanipulation eines Roboters via Internet mittels VRML2.0 und Java", Diplomarbeit, Institut für Robotik und Systemdynamik, Technische Universität München

[RL97] P.Ramos, F. Lobo Pereira, "Localisation system for an Autonomous Mobile Platform" Proceedings of the ISIE'97 - IEEE International Symposium on Industrial Electronics, Guimarães, Portugal, 7th-11th July, 1997.

[Saf96] Saffiotti, A., "Robot navigation under approximate self-localisation", *Robotics and Manufacturing Vol.6, Procs. 6th ISRAM Symposium, P589-594, 1996.*

[SC93] Scheile, B., and Crowley, J. L., "Certainty Grids: Perception and Localisation for a Mobile Robot", IRS '93, Zakopane, Poland, July 1993.

[TSW+96] Tittel, E., Scott, C., Wolfe, P., Sanders, C., (1996). Building VRML Worlds. Obsborne ISBN 0-07-882233-5, July 1996.

[VNET] VNET: http://ariadne.iz.net/~jeffs/vnet/

[Workspace] http://www.rosl.com/brochure.htm

An Integrated System for Managing Intelligent Buildings

T.Walsh, P.A. Nixon, S. A. Dobson

{tim.walsh,paddy.nixon, simon.dobson}@cs.tcd.ie

Abstract. As small wearable computing devices become more prolific it seems astute to fully capitalise on their potential to assume responsibility for a range of trivial tasks. The mobile phone which many people now own will undoubtedly evolve from the simple two-way communication purpose it fulfils today, into an integrated communications and organisation companion. We are interested in the integration of such smart devices into an intelligent building framework. Sensors and door activators are useful in their own right but to fully integrate them into a building management system requires the input from many diverse disciplines in engineering and computer science. This paper seeks to explore the technical issues of dealing with a system that manages the information flow through such a building. Specifically it examines the services that make up the core of the building's system.

Introduction

Miniaturisation of microchips continues to present the world with smaller computers of increasing power. Desktop power from the recent past becomes palmtop technology in the present and handheld or wearable in the future. There is a greater acceptance of modern devices as genuinely useful addition to everyday life. Wearable technology has been with us for several decades in the form of digital watches, with expensive models having diverse functionality. It seems likely that eventually the functionality of PDA's and mobile phones will be integrated into similar sized devices. Big brother theory aside these devices could be put to use as a form of identification and tracking should the user wish. This paper seeks to explore the problems of software control for in the area of smart areas and intelligent buildings. It demonstrates the motivation in section 2 and examines the implementation of a management architecture and supporting infrastructure in section 3.

Motivation

Buildings are having increasing amounts of automated systems installed to increase their reactiveness and function, and therefore improve aspects of usability and control in the environment. They are designed to operate autonomously with little or no user input. Building materials are also experiencing evolutionary jumps in their composition with remarkable abilities. Examples include special floor and wall materials which retain heat during cold spells but remove it when the outside temperature increases. Clever design and inventive materials are enabling energy savings to be made so that a building uses a fraction of the energy it would have used previously in heating and air conditioning.

The fact remains however that present building systems are typically based upon climate control, ensuring than the temperature, humidity, lighting and air quality are maintained at comfortable levels. These problems have been in environmental control domain for a prolonged period and have been more or less solved. A typical example of the attention paid to such detail can be found in the Architectural Brief [1] for the new library in Trinity College. Another case study [7] performed at the Lawrence Berkley National Laboratory demonstrated how buildings could be remotely monitored across the Internet from a single control centre. The system interfaced with multiple heterogeneous legacy building Energy Management Control Systems (EMCSs). Despite the comfort these systems provide and the transparent manner in which they do the task buildings have an enormous potential to provide more 'ease of use'. The extra independence afforded to disabled people can not be underestimated. Removing the dependence they have on able bodied people can improve self esteem and quality of life. Building controls could be all accessible from a common user interface on a small handheld device. A building which knows the limited mobility of an occupant (because of wheelchair access) could direct the user to a waiting lift instead of the unusable stairwell. An intelligent wheelchair could interface with the building allowing it to bring the occupant to their required destination. The lights in the person's office should switch on automatically and dim upon leaving with environmental controls set to the user's preference.

The extra ease of use is not limited to disabled people. Considering that most office workers spend a large part of their working life in a building it seems only fitting that that building should be as flexible and complimentary as possible. Gaining access to one's workspace should be an automatic event for the area's owner but prohibit unknown people entry. The Intelligent room project in MIT [4] provides two rooms with cameras for eyes and microphones for ears to make accessible the real-world phenomena occurring within them. A multitude of computer vision and speech understanding systems then help interpret human level phenomena, such as what are people saying and where they are standing. By embedding user interfaces this way, the fact that people, for example, then to point at what they are speaking about is no longer meaningless from a computational viewpoint, and systems have been built that make use of this information. Applying this technology in a boardroom allows a room's controller to tracks attendees, forwards non-emergency calls and takes minutes. Advances in sound capture allow [6] a network of microphones to pick up the speaker in a conference or lecture room as distinct from other noise sources

Part of the reason for this lack of progress is the lack of cross research in relevant research areas. Although conceptually simple the infrastructure required for such a system necessitates, at minimum, input from the follow key areas:

sensor development, vision and computer recognition, task user profiling (data mining to uncover patterns in behaviour), user interfaces, intelligent agents and distributed systems. The Intelligent Interfaces and Buildings (IIB) group in Trinity takes research work from the many different groups in college and attempts to provide an initial implementation of such a building.

To summarise, smart environments will perform planning tasks such as suggesting routes through the building to visitors, action tasks such as calling lifts, software tasks such as managing the massive event reporting such real-world systems will generate, and perception tasks such gesture and face recognition. All these complex interactions on such a large scale require an explicit supporting software architectures The remainder of this paper will concentrate on the software tasks involved in building such an architecture.

Implementation

A building [7] can be viewed as an entity made from many individual objects. The objects will be either fixed or mobile and can be broken down into the following categories:

- Fixed elements: These are items which are not computational or processing ability but will have state which can be ascertained using sensors. Examples include doors and windows (that can be open or shut), lights (on, off or dimmed), thermostat controls and even rooms themselves.
- Mobile elements: It is not necessary to install processors to everything that can move. Certain furniture and fittings do not have any use for it. Chairs and desks can be brought from one room to the other. Adding a magnetic tag, similar to those used to prevent theft, with a particular signature allow the building to track these types of elements throughout the premises.
- Fixed computing elements: This set of elements comprises of objects which have processing ability but no method of moving. The more obvious examples would include printers, computer terminals, coffee machines, photocopiers, air conditioners.
- Mobile computing elements: These objects exhibit the properties of fixed computing entities but also have the freedom of movement. Examples include mobile communicators (phone/personal organisers), intelligent wheelchairs, mobile personal computers and task specific robots.

These elements are the building blocks of an intelligent building. What is required is a software infrastructure which will handle all the interactions which will occur through typical building usage. We examine the task of handling all elements and all eventualities that can arise.

1.1. Mobility Architecture

The mobility architecture's task is to ensure that mobile computations can transfer from node to node successfully and continue executing their tasks. What objects end up being migrated is dictated by the decision of the policy service (Section 3.3) and event service (Section 3.2).

The structure of a generic migratory distributed application is displayed in figure 1. It shows how each component (also referred to as an autonomous object) can exist on a different node and yet remain in communication with other components of the application. Here the term node refers to a single processor, typically a single host. The circles represent components, while the dashed lines represent the various network references, or simply links, between them. The components are capable of being migrated to another node yet still carry on its computation, and more importantly its role in the computation. This is the goal for a real world system but prior to building such a system it is necessary to describe a disciplined abstract structure.

The complete structure of the architecture can be encapsuated in a logical database. It represents a rational method to store such a configuration because it is well structured, secure and retains data integrity automatically. The logical database can be implemented as a single, distributed or federated database, and we use the central database term solely to convey the methodology to be used in implementing such a system.

Different interfaces to the database allow access to the data. These interfaces are in the form of the standard services such as those specified by the OMG, RM-ODP, ODMG, and IETF. Due to the fact that they are stateless they represent logical filters to all data. For consistency we assume the definitions of the OMG for discussions of services named below.

The relationship service and naming service will assist migrating components to rebind to well specified or generic services. For example, should a migrating component wish to avail of a standard printing service on a destination host then the relationship service will provide the appropriate resource name. If the docking component has special 'secure' privileges then the relationship service may specify a secure output device such as the manager's printer while 'normal' components are referred to a standard office printer. Using this evaluated resource name a concrete binding to the particular resource is returned from the naming service. The Event Service [4] allows objects to communicate using decoupled event-based semantics. The Query Service provides an interface in much the same way as SQL provides an interface to databases.

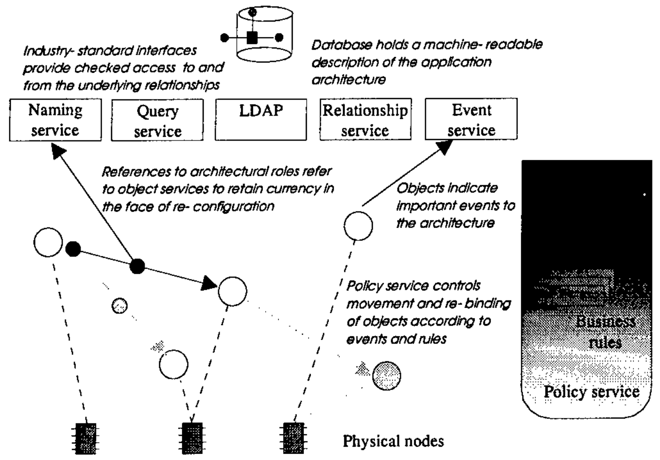

Industry- standard interfaces provide checked access to and from the underlying relationships
Database holds a machine- readable description of the application architecture
Naming service
Query service
LDAP
Relationship service
Event service
References to architectural roles refer to object services to retain currency in the face of re- configuration
Objects indicate important events to the architecture
Policy service controls movement and re- binding of objects according to events and rules
Business rules
Policy service
Physical nodes

Fig. 1. A Managed Architecture Mobile Distributed Applications

All the services see the same data in the database. Their purpose is to give the component a view of just the data needed to provide the bindings between resource names and their object references. This provides a seperation of data from the interfaces used to access it. Migrating components use the service interfaces to reconfigure their resource bindings. Each migration is different and can have different consequences. Use of standards allow the architecture to evolve as more interfaces become available over time and therefore, such an open system has the potential to do for migratory applications what OMG's CORBA did for remote object invocation.

1.2. Proposed migration technique

The research community has two broad notions in regard to mobile computations. Cardelli [9] states that mobile agents are meant to be completely self-contained. They do not communicate remotely with other agents, rather they move to some location and communicate locally when they get there. Agents are generally perceived to have an intelligence aspect which is lacking in simple mobile object systems. Mobile objects have their course set out from the start. They exhibit no independent judgement and can be though of as a drone. These methods represent opposite ends of the control spectrum. One having complete autonomy the other having none. We propose a mid grained approach where the key goal is **control and not constraint**.

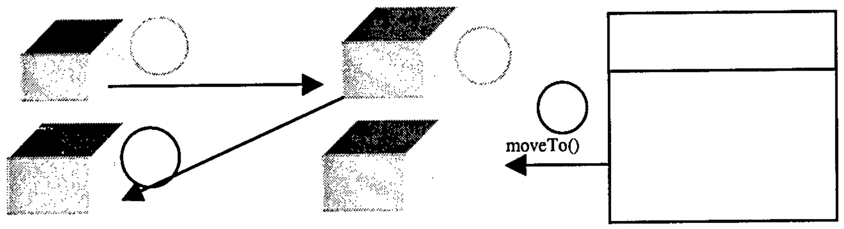


Fig. 2. Migration Technique

An important part of the architecture resides in the policy management unit. Each component, which is capable of migrating, has a 'moveTo()' method for such a purpose. The component itself can not call this method. Instead the policy unit invokes it, providing it with a destination and any other necessary parameters. The policy manager can dictate for example, whether it will allow a particular component access to its node or even whether an object can leave its node. Such decisions are dynamic and complex, and the design of a comprehensive policy manager is outside the scope of this paper. Were the component able to invoke moveTo() itself it would exibit the primary trait of an autonomous agent. Such autonomy can be simulated within the architecture by the use of a null policy which simply defers to requests of the requesting object. Policies are the key to adaptibility, as they also allow a range of different design choices to be facilitated in a single implementation. Policies even enable for design decisions to be revised post implementation without the necessarily having to change the algorithmic solution.

1.2.1. State Space Access

The Name Service in OrbixWeb [4] provides a standard OMG service. The format it saves its data in is proprietary. We have implemented a stateless name service which saves its data in an oracle database. This procedure allows other services to view the same data and alter it accordingly without causing conflicts which would arise were the name service stateful. Another advantage is the ease of use from a management point of view. It separates the concerns of users and administrators.

1.2.2. Architectural Pointers

Central to a stable migration is the abstract concept we refer to as an *architectural pointer*: a reference to an object specified relative to the architecture database by way of one or more of the services made available by the architecture. The pointer essentially encapsulates a query against the database together with the object resulting from the last time the query executed.

The power of architectural pointers comes from the use of the database. By combining architectural, configuration and placement data within a single logical structure, and allowing different views through the services, it is possible to identify roles in a very flexible manner. It also allows objects to reference roles directly, rather than purely retaining links to the objects which happened to fulfill those roles at any point in time. Finally, by acting as event consumers, they allow the policy component to re-configure the application and propagate these changes (in the form of pointer invalidations) directly to all affected objects.

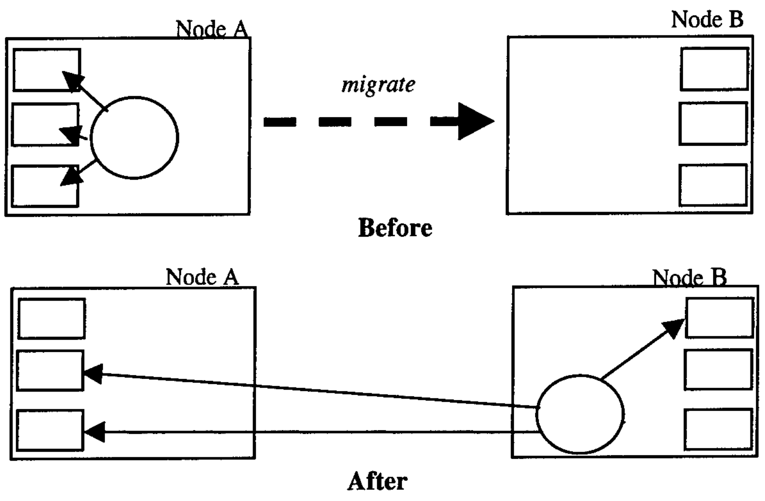


Fig. 3. Architectural Pointers

As an example of the utility of architectural pointers consider an object using both shared user database (D) and a printer (P1), with the constraint that the object always uses the same database irrespective of location while it's printer is defined as the printer on the local node. Architecturally this may be specified by identifying the user database explicitly by name and the printer in relation to the local node (figure 3 - before). If the object is migrated (figure 3 - after) then the database reference will remain valid but the printer reference will be invalidated, and when re-evaluated will re-bind to the printer on the object's new node (P2). The invalidation and re-binding is performed by the migration run-time system.

1.3. Event Service

On examination of a typical room in an office, depending on the level of granularity you choose, there can be many thousands of different things going on all around. Every time someone or something moves can be classified as a 'happening' or an 'event'. Typically in an office environment we are only interested in the events that interact on the human level. These are occurrences which we can see and observe. People moving from one room to another, doors opening and closing, a printer finished printing, a kettle boiled, these all fall into events which happen or which we cause to happen.

Even in a relatively small building and even at this level of granularity there are still immense amounts of information relating to the full set of events. Each object which raises an event wants all interested parties to know that this event has occurred. The simplest manner to deal with this is to broadcast to all who are listening. However, when examining all the objects which make up the average building, this will inevitably be to much information. What results is an 'event storm' where all objects are trying to tell every other object about it's condition. This leads to total congestion on the communication medium with the effect of nothing, or very little, getting done.

What is required is a system which will manage these events and control the flow of event notification only to those objects who require it. This is achieved by firstly categorising objects into sources and sinks. An object is an *event source* when it generates events. Therefore, a door opening is a type of event so the door object is an *event source*. An *event sink* is an object which is interested in particular events. An example here might be the security object, which is interested in a the door object being opened. It will want to check whether the room should be accessible and whether the person entering it has permission to do so. For the *event sink* to be informed of such events it needs to register with the *event source* requesting that it send it an such occurrences.

1.3.1. Event Service Advertising

As mentioned previously an object which produces events are referred to as *event sources*. Prior to the issuing of events objects have first to advertise their intention to publish events.

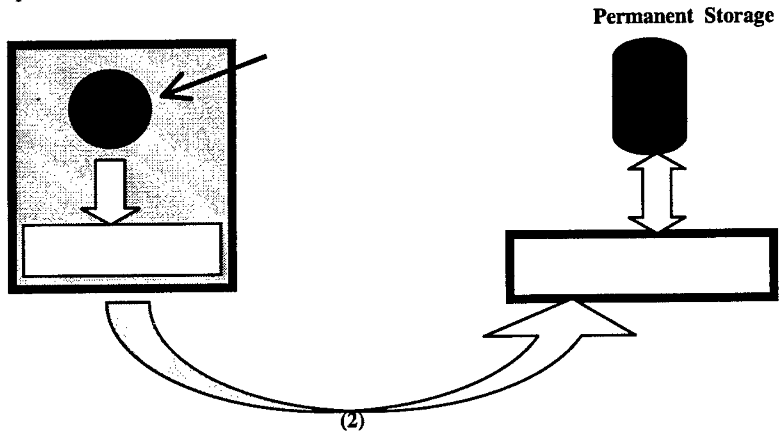


Fig. 4. Event Service Advertising

The first step in advertising an will be a call to the *Advertise(name, other info)* method to indicate to the event bus it readiness to produce events. A naming system must be adopted whereby the events can be uniquely identify throughout the system. To this end a proprietary naming scheme will be used for the naming of the events.

An example URI that could uniquely identify an event might look like the following, *ie.tcd.Oreilly.floor2.Simon.enterEvent*. This would uniquely identify the event, but also give the location to where the information on the event is found and what protocol is to be used.

The event bus is the support mechanism for production and consumption of events. In this model an event bus will be located on each device. The Advertise method passes the event bus the name of the event, which is the identifier discussed earlier. It also passes other information such as owner event source.

The next step is for the local event bus to forward this information, along with location of the event source, onto the Event Naming. This is the central repository for named events. It is a look up mechanism which *event sinks* use to discover an *event source*. After the Naming Service receives the information, it inserts it into its database thereby adding the event source to the list of other sources. Once this is completed the event source is considered as been advertised.

1.3.2. Event Subscription

As with most event services clients are required to, in some way, subscribe to events that they are interested in. This narrows the scope of the events on the overall system. It also prevents event storming by only sending the notification of the event to the parties which have shown an interest in that event. Clients in this event service are required to subscribe to an event using the full name of the event

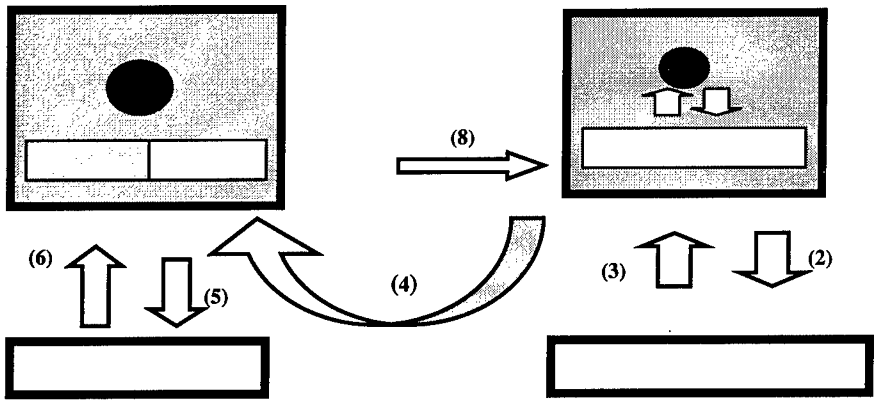


Fig. 5. Subscribing to Events

Figure 5 shows the series of steps required for a client to subscribe to an event. There are four distinct stages to subscribing; the client making the request, finding the location of the event bus which support the event source, subscribing to the event bus and the installation of the filter. The filter in this case is acquired from the policy service which is discussed in Section 3.3. Suffice to say that the filter will ensure that event sinks will only receive notice of the event if certain eventualities prove true.

An event sink can unsubscribe from event sources. It does so in a similar, but reverse order, fashion as event subscription.

1.3.3. Event Notification

Figure 6 shows the manner in which the event bus informs registered objects. The event source notifies it local event bus of a new event and passes the parameters associated with this instance of the event. The job of the event bus is to then to notify the interested parties of the event.

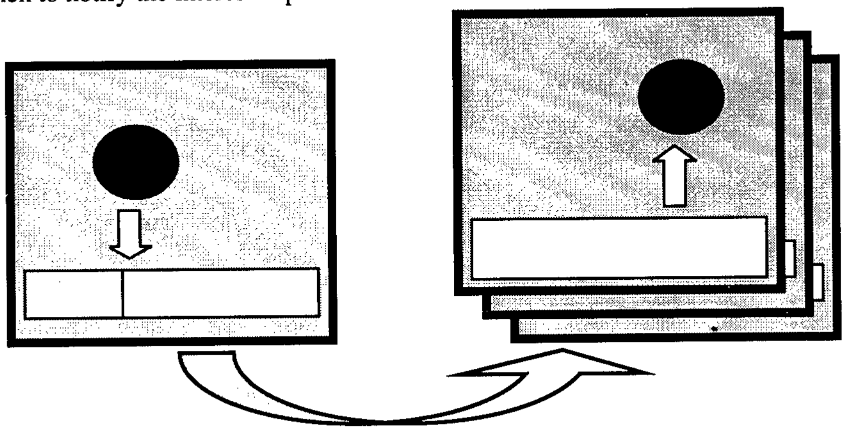


Fig. 6. Event Notification

1.4. Policy Service

All formal organisations have policies, which are defined as 'the plans of an organisation to meet its goals'. Policies, on a more abstract level, represent a capability or a capacity to do something.

Take the example of a typical building, be it a shop, an office or a home environment, policies are in force. Offices and shops typically have a security guard with specific instructions from a building manager. They will only allow certain authorised individuals into sensitive areas of the premises or they may refuse others access altogether. In this particular case the person attempting to enter is referred to as the policy subject while the security guard is the policy target. The target will have actions available which are to be accessed by the subject. In this example the only action available might be *enter*. If there is a policy constraint associated with the target object, for example *"are underage"* then the *enter* action may not be accessible to a person who is declared to be underage.

The primary purpose of the policy service is to model these rules in such a way that they are specific, unambiguous and fulfil their purpose.

To model a building for policy needs, a tree of zones is mapped out. Figure 7 shows the O'Reilly Institute in Trinity mapped out into constituent zones. The fixed elements which a building contains (refer to section 3) will be assigned to specific zones while the mobile elements have the ability to move between zones. Zones and objects have policies associated with them. For all elements in the environment a policy object will be associated with it.

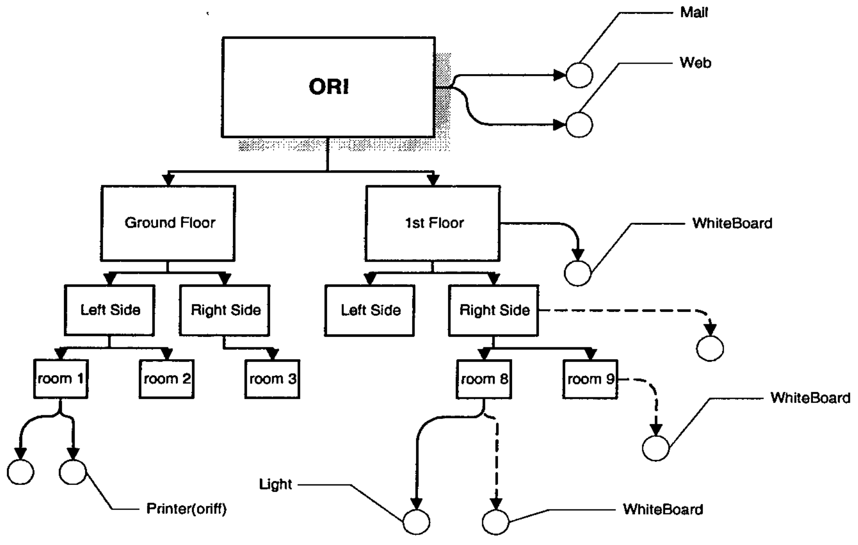


Fig. 7. Zone tree

As mentioned in the example above policies have certain elements associated with them. A full definition is provided by Sloman. According to Sloman [2] a policy can be defined by five criteria.

- **Modality:** A policy may be authorisation or obligation. In both cases the modality can be positive of negative. Positive authorisation (A+) is a flag which indicates that if a certain action occurs the perpetrator will be permitted to carry on. For example a system administration person attempting to enter a server room would be allowed. Negative authorisation (forbidding) provides the opposite effect so that a policy could be installed to withhold access for all non system administration personnel unless they are accompanied by an authorised person.

Positive obligation (O+) ensures that if a particular action takes place then this policy dictates a course of action which must be followed. An example could be that if the non sys-admin person was in the server room then that presence requires a log entry in a security database. Negative obligation provides a deterring mechanism

- **Policy Subject**: This attributes defines the user objects to whom the policy applies. The policy subject can be a specific user, a group of users, or a list of users and groups.
  PS: Simon
  PS: All
  PS: Under, Post, Paddy

The policy subject is classified as the producer of an event. The full relationship between events and policies is discussed at the end of the section.

- **Policy Target Object**: it defines the objects at which the policy is directed. It can also describe a list of objects or a zone. The target object is the consumer of events.
  Printer_oriff
  Mail_Server
  Door_G51

- **Policy Action**: This attributes defines the method to which the policy applies. This represents a specific method of the policy object. These methods are available to the policy subject providing that no constraints are in place. On the pretence that no constraints are in place, the policy subject can invoke those methods on the target object.
  Print()
  Setup()
  Send_Mail(Mail)

- **Policy Constraints**: Constraints serve as our way of setting rules. They give us control over the policy subject's actions. The basic structure is a set of Boolean methods that must all return. Failure to do this will prevent access to the target's functionality. The conditions are very specific to the object. No constraints means free access. This constraint can be another policy. In that case the policy in question have to be respected to fulfil the present one. The following scenario gives an insight as to how constraints work. Simon's office door has a door opening mechanism. The door is the target object. Associated with this is it's action *openDoor*. Also associated is the constraint *Access(UserID)*. As Simon (policy subject) approaches the door an event is generated. The *openDoor* action is not immediately available because a constraint is in place. The UserID is passed to the policy, it returns true as Simon has full access so now *openDoor* is invoked and the door sweeps open. All this information is encapsulated in a policy object.

Cond (Job_Name)	Allow to print only a certain job

Cond (Time)	Allow only at certain time

- **Owner**: The Owner attribute specifies the owner of the policy. It is used for allowing changes to the policy and for accounting or requests such as all the retrieval of all the policies of a specific user. By default the owner of a policy is the application that owns the object, but the administrator can always modify policies.
  Owner: Admin
  Owner: Printing_Service
  Owner: Paddy

Policies, events and system management are all tightly related. The event service generates the events, the policy service defines the capabilities of all entities/objects and the management service carries out the required tasks

1.5. Current State of Implementation

The current state of implementation is as follows. The event service, policy service and migration architecture have been implemented. The state of the system is stored in the computer science's oracle database and is presently undergoing tests. The O'Reilly institute is soon to be wired with sensors and detection devices which will allow wireless communication and interaction with the building. No critical or secure systems will be placed on-line until full beta testing of the primary system is complete.

The next phase planned is to find a suitable real-world location as a test bed for the system. An ideal location would be a typical office setting which would be beneficial in the search for more problems which did not occur on the smaller scale of the O'Reilly Institute. After the scalability and reliability of the system has been proven in real world conditions it will be ready for deployment in more critical sites such as convalescent homes and disabled training centres.

1.6. Related Work

Although there have been several implementations of event services, policy services and mobility frameworks there has been no concentrated effort to combine them into a useful system for the intelligent building domain. Jini [3] is Sun's new technology which sits on top of Java and provides seamless interconnection between all devices be they DVD players, air conditioners or cellular phones. Lucent's Inferno [8] also provides a similar task.

Conclusions

Maturing technologies, in both relevant hardware and software domains, has led to a re-emergence of interest in living and working in an environment that adapts to users needs and requirements. We have presented an aspect of that concept. The system that supports the management of all objects in a scalable manner is the essential backbone. In terms of the mobility aspect of the system it represents a mid-grained mobility approach. In contrast to the coarse-grained mobility, which is based upon intelligent agents that operate without user interference or input, and the fine-grained approach that takes the autonomous object approach, requiring total interaction with a user or control sub-system. From this mid-grained approach we gain control but do not constrain the system.

The event and policy services available to the mobility system give the allow people, computers and inanimate objects to interact in a useful manner

Acknowledgements

The authors wish to acknowledge the help of Peter Barron and Thomas Kunetz in the preparation of this paper.

References

[1] Trinity College Library, New Library Building, Architectural Brief
available from http://www.tcd.ie/Library/Local/Newlib/Brief/

[2] J.D. Moffett, M.S. Sloman, "The Representation of Policies as System Objects", SIGOIS Bulletin, Vol. 12 No.2, pp 171-184.

[3] Jini Technology Executive Overview, Sun Microsystems, Inc.
available from http://www.sun.com/jini/overview/

[4] Michael H. Coen, "The future of human-computer interaction, or how I learned to stop worrying and love my intelligent room", IEEE Intelligent Systems, March/April 1999.

[5] Michael C. Mozer, "An intelligent environment must be adaptive", IEEE Intelligent Systems, March/April 1999.

[6] James L.Flanagan, "Autodirective sound capture: towards smarter conference rooms", IEEE Intelligent Systems, March/April 1999.

[7] Frank Olken, Hans-Arno Jacobsen, Chuch McPartland, Mary Ann Piette, Mary F. Anderson, "Object Lessons Learned from a Distributed System for Remote Building Monitoring and Operation", ACM SIGPLAN, Vol 33, No. 10, October 1998.

[8] Sean M. Dorward, Rob Pike, David Leo Presotto, Dennis M. Ritchie, Howard W. Trickey, and Philip Winterbottom, The Inferno Operating System, Lucent Technologies.

[9] Luca Cardelli, "Mobile Computation", Microsoft Research. *available from* http://www.luca.demon.co.uk

PROGRAMMING SUPPORT

Real + Virtual = Clever
Thoughts on Programming Smart Environments

Mads Haahr[1], Vinny Cahill[1], and Eric Jul[2]

[1] Department of Computer Science, Trinity College, Dublin 2, Ireland
{Mads.Haahr,Vinny.Cahill}@cs.tcd.ie
[2] Department of Computer Science, University of Copenhagen, Universitetsparken 1, DK-2100 København Ø, Denmark
eric@diku.dk

Abstract. Event-based communications has been used successfully in many application domains, one of which is virtual environments. Events are a useful concept in this context because they embody the notion of something happening in an environment, be it real or virtual. We claim that the notion of events is not only a suitable communications paradigm to model purely virtual environments but that it can also be used to *interface* an area in a real environment with a corresponding area in a virtual environment by relaying events in both directions. This idea, in turn, could turn out useful as a programming model for smart environments. As part of our recent work in distributed event systems for virtual world support, we have implemented an event model called ECO and used it to build a virtual model of a real world environment. In this position paper, we describe how such a virtual environment can be interfaced to its real world counterpart. We argue that this technology is promising as a way of programming smart environments because it maps naturally onto the application domain and simplifies the programming of such environments. To support this view, we present some examples of functionality often found in smart environments and explain how they can be programmed with the technology presented in this paper.

1 Introduction

Event-based communications has been used successfully in many application domains, one of which is virtual world support [5]. Events are a useful concept in this context because they embody the notion of something happening in a world, be it real or virtual. We claim that the notion of events is not only a suitable communications paradigm to model purely virtual worlds but that it can also be used to *interface* one or more areas in the real world with a corresponding area in a virtual world by relaying events in both directions. This idea, in turn, could turn out useful as a programming model for smart environments.

This position paper describes how event-mapping can be done and presents an application which can map a single type of event between a real and a virtual environment. We argue that this technology is promising as a way of programming smart environments because it seems to map naturally onto the application domain and make the programming of such environments simpler. To support this view, we present some examples of functionality often found in smart environments and explain how they can be programmed with the technology presented in this paper.

2 The ECO Model

The ECO Model is an event model developed for distributed virtual world support. It was used in the Moonlight [4] project (a distributed 3D video games project) and, as event models go, it is relatively simple. It has only three central concepts and its application programmer interface (API) contains only three operations. The intent of the model is that it is applied to a given host language and extends that language's syntax and facilities so as to support the ECO concepts. Though the ECO model has traditionally been used in a virtual world context [8, 6], the model itself is generic, and can easily be used in other domains where event-based communication applies. This section describes the ECO concepts and operations.

2.1 Events, Constraints and Objects

The acronym ECO stands for *events, constraints*, and *objects*—the three central concepts in the event model:

Objects in the ECO model are much like objects in a standard object-oriented language. However, instead of invoking other objects for communication ECO objects communicate with other ECO objects via events and constraints as explained below. ECO objects are often implemented as programming language objects but not all programming language objects are necessarily ECO objects. In order to distinguish the two, ECO objects are often referred to as *entities.* Entities have identifiers that are unique within an ECO world and they may contain threads of control.

Events are the only means of communication in the model. Entities do not invoke each other's methods directly but instead raise events which may, or may not, lead to other entities' methods being invoked. Any entity can raise an event. Events are typed and have parameters, and they are propagated asynchronously and anonymously to the receiving entities in no particular order. The type of events is usually specified using the type system of the underlying language.

Constraints make it possible for entities to impose restrictions upon which events they actually receive. The ECO model specifies several types of constraints: *pre, post, synchronisation*, and *notify* constraints. Notify constraints (known as *filters* in some event models) can be used by an entity to specify what events it is interested in receiving *notification* about.

The three concepts are illustrated in figure 1 which shows two ECO entities communicating. Entity A raises an event which may, or may not, reach entity B

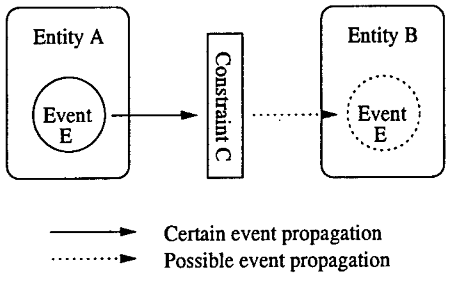


Fig. 1. The Three ECO Concepts in Relation

because of the constraint C. The constraint is imposed by entity B. The raising of an event can be thought of as an announcement to the rest of the ECO world that the event has occurred. A notify constraint can be thought of as a filter that decides whether or not a given entity is to receive the event, and receiving an event can be thought of as invoking an appropriate method (called an *event handler*) of an entity in response to the event. When an entity uses a notify constraint to enable it to receive certain events, we say that it *subscribes* to those events. An entity can subscribe multiple times to the same events using different constraints and handlers. It is also possible to subscribe without using a constraint, in which case no filtering is performed.

2.2 The ECO Model's API

The ECO model's API contains three operations which are used by entities to communicate:

`Subscribe(event-type(notify-constraint), event-handler)` is used by entities to register interest in events. An entity that subscribes to a certain type of event will receive an invocation of one of its methods when a matching event is raised. The event is passed as a parameter to the handler. When an entity performs a subscription, it can also choose to specify a constraint. An event must be of the right type and must match the constraint (if any) to be delivered to a particular subscriber.

`Raise(event)` is used by an entity to announce the occurrence of an event. The event is delivered to all entities subscribing to events of that type, subject to filtering against their respective constraints.

`Unsubscribe(event-type(notify-constraint), event-handler)` is used by an entity to cancel an existing subscription.

3 The Real and the Virtual Environment

As mentioned in section 1, our environment consists of a real and a virtual part. This section presents the two halves of this environment and the next discusses how events can be mapped between them.

3.1 The Real Environment

The real environment is a series of research laboratories found at the University of Cambridge. The laboratories are equipped with specialised hardware called an active badge system. The badge system consists of a number of infrared sensors (called stations) placed in the rooms and hallways of the university buildings. The stations pick up signals emitted by battery-driven badges worn by personnel in the labs. When a station detects the presence of a badge, it raises a so-called sighting event to announce that this particular badge has been seen in that particular location. Stations are grouped into networks, each being a part of a particular laboratory. Each badge carries a unique badge identifier which is picked up by the sensors.

Because we do not have access to the actual badge system, the events raised in the virtual environment are replays of events that happened at an earlier point in time. The data we have obtained from the system consists of 35,811 sighting events collected over period of almost 21 hours by 118 stations distributed over 12 networks. It is worth noting that the virtual environment is run as a simulation (i.e., using previously collected data rather than real-time data) only because we do not have access to the data-generating hardware. Given the required hardware, the virtual environment could easily be maintained in real-time. For each sighting, the following information is available:

Location Identifier identifies a physical location in the environment. The location identifier consists of a *network* (a symbolic name) specifying an area of the building and a *station* (an integer) specifying a location within that area. Assuming there is only one station in each physical location, the location identifier can be mapped uniquely to a location in the virtual world.

Badge Identifier identifies the sighted badge by a sequence of six eight-bit hexadecimal numbers separated by dashes. This identifier can be mapped to the real name of the badge owner.

Time Stamp identifies the moment when the sighting was made in seconds and microseconds, since 00:00:00 UTC, January 1, 1970, as returned by the `time(3)` Unix system call.

3.2 The Virtual Environment

The virtual environment is a model of the real environment and forms a shadow world where events happening in the real world are echoed. For each station in the real laboratories there is a corresponding location in the virtual environment and for each badge identifier there is an entity in the virtual environment representing that person. The entities are moved between locations in the virtual environment as sighting events are raised by the stations.

The station entities in the virtual environment represent physical pieces of equipment and echo real events by raising virtual (ECO) events. In addition to the station entities, the virtual environment also contains entities with no physical counterparts. These entities can interact with each other and the rest of the virtual environment by using the three ECO operations to subscribe to events (such as those raised by the station entities) and raise events of their own. Below is a description of a series of such entities.

Location entities implement rooms in the virtual environment. For each station there is a corresponding location entity keeping track of which entities are currently in that location. This location entity subscribes to all sighting events from the corresponding station in order to detect arrivals of new entities. Entities are assumed to remain in the location where they were last sighted until they are sighted elsewhere. For this reason, a given location entity also subscribes to all sighting events featuring badge identifiers which are currently in its own location. A location entity can be queried (using an ordinary non-ECO invocation) as to who is currently in that location.

Ghost entities implement people in the virtual environment who have no physical counterparts in the real environment and therefore do not cause the sensors to raise sighting events. A ghost entity is controlled remotely by a user using a text-based command line interface. Ghost entities interact with location entities in order to move around and to present their remote users with views of their current locations in the form of a textual descriptions.

CCTV (closed circuit television) entities implement security cameras in the virtual environment. A CCTV entity subscribes to all sighting events occurring within a particular network and in this way monitors a small area of the entire virtual environment.

4 Interfacing the Real and the Virtual Environment

This section discusses a possible mapping of events between the two environments by looking at how one type of event, the *sighting* event, can be mapped in both directions. In general, it is important to distinguish between the representation and the presentation of a virtual environment. The former is the way the environment is structured and stored in the computers' memory and is independent of any input/output devices individual users may have. The latter is the way the environment is presented to individual users and is therefore dependent on such hardware. In particular, it is possible to have the same virtual environment presented differently to different users.

There are many ways of presenting virtual environments to users, ranging from simple text-based interfaces as those used in MOOs (Multi-User Environments, Object-Oriented) [1] over first person perspective 3D graphics commonly used in popular games such as Quake, to full-blown virtual reality equipment such as stereoscopic VR goggles or even CAVE theaters [3].

For the purposes of this paper, we use the simplest possible presentation: a text-based interface akin to that used in MOOs. It is important to stress, however, that the ideas discussed here are general and will be usable with other presentation techniques. It should also be noted that researchers from the Computer Laboratory at the University of Cambridge are doing work which is somewhat related to that presented here. Their approach is to use the active badge system to maintain a VRML2 representation of the labs. This enables users to interact with the environment using a graphical rather than a text-based interface [2].

It should also be noted that this is work in progress and most of our effort so far has been to implement an application that can map real to virtual sighting events. Therefore, the discussion of real-to-virtual event mapping is more thorough than that of virtual-to-real mapping.

4.1 Example

The following is an example of a purely virtual user interacting with the environment. At the user's location is another virtual user, John, and a real person, Jane. Jane herself is physically present at the corresponding location in the real environment whereas the two virtual users are only present at the virtual location. User commands issued by the virtual user from whom the transcript is taken are preceded by a prompt (>) character.

```
You are on the 3rd floor of the Research Laboratory.
A sign on the wall reads, 'Area 8.'  A sensor in the ceiling is marked, '11.'
John (virtual) is here.
>look at john
John is transparent and has no badge identifier.
Jane (real) arrives.
>look at jane
Jane's badge identifier is 0-0-0-0-10-14.
```

There are several things to note about this example. First, John and Jane are marked as virtual and real mainly for reasons of clarity. Distinguishing between users in this way is not a strict necessity. Second, John has no badge (he is purely virtual) and therefore no badge identifier. Third, this scenario is limited because it features only one type of event: the sighting event. Though this event enables users to see each other and move around (by being sighted in different places) it offers few possibilities of interaction.

4.2 Sighting Events

In the example, the location entity keeps track of any real and virtual users present at the location. When Jane arrives physically at the location, the signal emitted by her infrared badge is picked up by the sensor in the ceiling and a sighting event is raised. This event is detected by the location entity because it subscribes to all sighting events raised by that sensor. Hence, Jane's arrival in the physical location of the real environment triggers her virtual arrival in the corresponding location of the virtual environment. This is effectively a mapping from a real to a virtual sighting event.

For sighting events, the purpose of a virtual-to-real mapping is to let Jane in the real location obtain a sighting of a virtual person in the virtual location. (If the person is real rather than virtual, we assume Jane will be able to see him with her own eyes.) This is a presentation issue and can be adressed in many different ways. In case the location is Jane's office and she has a terminal on her desk, a simple terminal window could be used to inform her about the presence of virtual users in her virtual office. If the location is a hallway or meeting room, a screen (possibly flat and wall-mounted to take up as little space as possible) could be placed in the location and messages about the presence of virtual entities could be either printed (MOO-style) or drawn (as images) on the screen. More ambitious approaches could use more advanced hardware to give a more lifelike presentation of events in the virtual world. For example, a holographic projection of each virtual entity would be ideal but is hardly feasible with current technology. Comparable results, however, might be achieved by equipping Jane with a wearable rendering computer and a pair of transparent VR goggles where an image of the virtual environment is superimposed that of the real environment. Regardless of whether a low- or a high-tech approach is adopted, the presentation would let Jane observe events happening in the virtual world. Analogously, Jane's badge would let virtual people see her. Together, these two mappings constitute a full mapping of sighting events in both directions between the environments.

5 Smart Environments

The previous sections have described a real and a virtual environment and explained how they could be closely interfaced by mapping events in both directions. So, the next question is, in what way is this technology useful for smart environments research? A smart environment can be seen as an 'ordinary' real environment with certain extra functionality that enables it to *monitor* its own state and to *modify* that state. When a smart building detects the presence of a disabled person in front a door, the detection is a result of the building monitoring its own state. (We here assume people in the building to be part of the environment.) When the building opens the door to let the person through, it is modifying that state.

Sections 3 and 4 argued that such state can be naturally represented as a virtual environment mirroring the the real environment and that *events* can be used to keep the two environments synchronised. In fact, event mapping is just another word for monitoring and modifying the real environment's state. When events are mapped from real to virtual, the real environment's state is being monitored. When they are mapped from virtual to real, the state is being modified.

The section 4.1 example used an extremely simple environment, featuring only surveillance-type functionality and using only a few types of entities and one type of events. *Surveillance* is only one area in which this technology can be applied. In the following, we discuss this and two other areas known from smart environments, namely *planning-* and *action-type* applications. The example from section 4.1 is extended to illustrate the points, and actual C++ code with ECO statements is used to explain how the events are used. It is worth noting that this section is work in progress and because the ideas are largely untested, the discussion is somewhat speculative. The code given in this section assumes the existence of a sighting event class along the lines of that given below.

```
class SightingEvent : public Event {
  string badge_id;  // The badge seen.
  string network;   // Location identifier:
  int station;      // (network, station).
};
```

5.1 Monitoring and Surveillance

Many 'dumb' buildings are equipped with security cameras to monitor areas which are deemed sensitive for one reason or another. Such cameras generally have the disadvantage that they have to be monitored by people rather than software because detailed digital image analysis is not only extremely difficult but also very time-consuming. Consequently, the information that can be gathered from such cameras is in practice rather limited. Areas are often monitored only for security purposes; other information, such as the movement patterns of (legitimate) users, is not recorded.

Smart environments can incorporate user tracking, for example via active badges as those described in section 3.1. This makes it easy to analyse the movement patterns of the users of the environment. The CCTV camera from section 3.2 is an example of a simple analysis tool, monitoring a small area of the complete environment. The following C++ code with ECO statements shows how a simple CCTV camera could be implemented; it subscribes to all sightings in a particular area and logs them to standard output. Recall that the *network* and *station* constitute a location identifier and that a *network* consists of a series of (related) locations. Asterisks ($*$) denote wildcards, i.e., fields matching any value.

```
class CCTV {
  string network;
  int station;

  CCTV(string nw) : network(nw) {
    // Detect all users in this area ('network') of the building.
    Subscribe(SightingEvent(*, network, *), &EventHandler);
  }
  ~CCTV() {
    // Cancel subscription.
    Unsubscribe(SightingEvent(*, network, *), &EventHandler);
  }
  void EventHandler(SightingEvent se) {
    // Log the sighting to stdout.
    cout << "Badge " << se.badge_id << " seen.\n";
  }
};
```

Entities like this CCTV camera are useful because the complete event flow through any sizable smart environment will be large and difficult to comprehend. Entities such as this can be used to provide meaningful views of this event flow by dynamically extracting events according to certain patterns, and in this way make it easier for humans to monitor the system at runtime. The CCTV camera extracts sightings on the basis of their *location* but other entities could for example monitor certain *users*, or use a combination of the two.

Using smart environments with user tracking doubtlessly has its advantages but also raises serious ethical issues about privacy. It is easy to picture technology like that described here being used to implement surveillance of Orwellian [7] proportions on the grounds that it is for the users' own good.

5.2 Planning

Planning routes through buildings is a useful functionality for visitors and robots and one that is often discussed in the context of smart environments. To make a plan from one location to another, a building must be aware of its structure, e.g., must know the layout of floors and the locations of lifts, stairs and doors. It must also take into account the abilitites of the user (be it a visitor or a robot), e.g., whether he/she/it can use lifts and stairs and holds the right keys and access codes to doors on the way. Even after a suitable plan has been laid out, a smart building may still help the user follow it for example by displaying encouraging messages on screens on the way.

Making the plan is an algorithmic problem and beyond the scope of this paper. However, once a plan has been made, the environment can assist the user in carrying it out. In the context of the section 4.1 example, this could be implemented with virtual *signpost* entities placed in locations along the route. Consider the following version of the example:

```
You are on the 3rd floor of the Research Laboratory.
A sign on the wall reads, 'Area 8.'  A sensor in the ceiling is marked, '11.'
Jane (real) arrives.
A green arrow bearing the name 'Jane' in large, friendly letters suddenly
appears on the wall.  It is pointing north.
Jane (real) goes north.
The green arrow vanishes.
```

This example features a signpost visible in the virtual world. Obviously, signposts would also have to be visible in the real world, in order to be useful for real users. For human users, they could be printed on displays in the various locations. For robots, they could be transmitted via short-range wireless links such as infrared. Note that directions only appear when the user enters the location and that they disappear when the user has left. This prevents an environment with many visitors from becoming cluttered with messages. Also, in the case of human users, chances are that the appearance of the message will attract the user's attention at the right moment.

The following C++ code with ECO statements shows how signposts could be implemented. A series of signposts could be created by a planning application and placed in the environment along the projected route. The exact placement of signposts is of course open to discussion; one could for example claim that a user does not need signposts when following the projected path, only when straying from it. However, for the purposes of this example, we will assume they are placed in the locations on the path the user is expected to follow.

```
class SignPost {
  string badge_id, network;
  int station;

  SignPost(string bid, string nw, int st)
    : badge_id(bid), network(nw), station(st) {
    // Start looking for our user in our location.
    Subscribe(SightingEvent(badge_id, network, station), &Activate);
  }
  // Activation Handler.
  void Activate(SightingEvent& se) {
    // User is here; become visible (code not shown) and
    // stop looking for the user in our location.
    Unsubscribe(SightingEvent(badge_id, network, station), &Activate);
    // Start looking for the user anywhere.
    Subscribe(SightingEvent(badge_id, *, *), &Deactivate);
  }
  // Deactivation Handler.
  void Deactivate(SightingEvent& se) {
    if (se.station != station || se.network != network) {
      // User is sighted somewhere else; stop looking.
      Unsubscribe(SightingEvent(badge_id, *, *), &Deactivate);
      // Destroy ourselves (code not shown).
    }
  }
};
```

During the lifetime of a signpost entity, each of the above methods will be invoked once in the order of *constructor*, *activation handler* and *deactivation handler*. The two handlers will be invoked when the appropriate sighting events (matching the subscriptions) occur. The signpost example shows how a simple guidance system can be implemented rather easily given a virtual representation of the real environment and mapping of sighting events in both directions.

5.3 Action

Another function of smart environments is to perform certain *actions* automatically and on user request. Common examples are calling lifts and opening/closing doors in the environment. This type of functionality is useful for users with limited physical abilities, such as disabled people and robots, but also for users who need to control certain aspects of the environment remotely.

Consider a smart environment with automatic doors that can be opened by anybody with the right key. When opened, a door stays open for 30 seconds whereafter it closes. In the virtual environment, doors are represented as ECO entities and keys as strings. A door can be attempted opened by raising a `PleaseOpenDoorEvent` in the virtual environment. This event contains the key (placed there by the entity raising the event) and is received by the door entity. If the key matches, the door opens. This is signalled by the door with a `DoorOpenEvent`. When the door closes, it raises a `DoorCloseEvent`. Each door has a unique identifier in the form of a string. C++ code for the three event types is given below.

```
// Raised by somebody who wants to open a door.
class PleaseOpenDoorEvent : public Event {
  string door, key;
};
```

```
// Raised by the door when it opens.
class DoorOpenEvent : public Event {
  string door;
};
// Raised by the door when it closes.
class DoorCloseEvent : public Event {
  string door;
};
```

Note that ECO events are global; there is only one event space where all events are raised. Hence, anybody can raise a `PleaseOpenDoorEvent` regardless of their location; in particular, it is not necessary to be anywhere near the door. In practice, the `PleaseOpenDoorEvent` could be raised either by actual users (when standing in front of the door), by security personnel in remote locations or by other parts of the building, such as the signpost entities described in section 5.2. The code for the door entity is given below.

```
class Door {
  string my_id, my_key;
  bool open;

  Door(string id, string key) : my_id(id), my_key(key) {
    // Subscribe to all request to open us.
    Subscribe(PleaseOpenDoorEvent(my_id, my_key), &Handler);
  }
  ~Door() {
    Unsubscribe(PleaseOpenDoorEvent(my_id, my_key), &Handler);
  }
  void Handler(PleaseOpenDoorEvent& ev) {
    // Rudimentary security.
    if (ev.key == my_key) {
      // Physically open door (code not shown) and raise
      // event to tell the environment it is now open.
      Raise(DoorOpenEvent(my_id));
      // Wait a while to let people pass through.
      sleep(30);
      // Physically close door (code not shown) and raise
      // event to tell the environment it is now closed.
      Raise(DoorCloseEvent(my_id));
    }
  }
};
```

This example shows how event-based programming combined with a virtual representation of the environment gives a high degree of flexibility in controlling and programming the building. For example, a remote control button for the door is easily implemented as an entity that raises the appropriate `PleaseOpenDoorEvent`. It can be placed anywhere in the building and even moved at runtime.

6 Conclusions and Future Work

This paper has presented ideas for closely interfacing real and virtual environments by mapping events between the environments in both directions. We have also described an ongoing implementation of these ideas which features only a single type of event, the sighting event, and only the simplest possible presentation layer, a text-based interface. Despite these limitations, we claim that the principles generalise and will remain valid in more complex scenarios with many event types and multiple (and more advanced) presentation layers. We have also briefly described the ECO event model which the implementation uses for event-based communication.

We have argued that this technology can be applied in the context of smart environments by giving examples of some typical applications from the domain and explained how they could be implemented with the technology. The examples suggested that at least some smart environment applications could be kept quite simple even for large environments. Our conclusion is that closely interfaced real/virtual environments is a promising programming model for smart environments and is worth investigating further.

Acknowledgements

The authors are very grateful to John Bates of AT&T Laboratories, Cambridge for supplying the data from the active badge system.

References

1. R. A. Bartle. Interactive Multi-Player Computer Games. Technical report, MUSE Ltd, Colchester, Essex, UK, 1990.
2. John Bates, Jean Bacon, Ken Moody, and Mark Spiteri. Using Events for the Scalable Federation of Heterogeneous Components. In *Proceedings of the Eighth ACM SIGOPS European Workshop*, September 1998.
3. C. Cruz-Neira, D. J. Sandin, T. A. DeFanti, R. V. Kenyon, and J. C. Hart. The CAVE: Audio Visual Experience Automatic Virtual Environment. *Communications of the ACM*, 35(6):65–72, June 1992.
4. TCD Team Moonlight. VOID Shell Specification. Project Deliverable Moonlight Del-1.5.1, Distributed Systems Group, Department of Computer Science, Trinity College, Dublin 2, Ireland, March 1995. Also technical report TCD-CS-95-??, Dept. of Computer Science, Trinity College Dublin.
5. Karl O'Connell. *System Support for Multi-User Distributed Virtual Worlds.* PhD thesis, Trinity College, Department of Computer Science, Dublin 2, Ireland, October 1997.
6. Karl O'Connell, Tom Dinneen, Steven Collins, Brendan Tangney, Neville Harris, and Vinny Cahill. Techniques for Handling Scale and Distribution in Virtual Worlds. In *Proceedings of the Seventh ACM SIGOPS European Workshop*, pages 17–24. Association for Computing Machinery, September 1996.
7. George Orwell. *1984.* New American Library, 1990. ISBN 0-451-52493-4.
8. Gradimir Starovic, Vinny Cahill, and Brendan Tangney. An Event Based Object Model for Distributed Programming. In John Murphy and Brian Stone, editors, *Proceedings of the 1995 International Conference on Object Oriented Information Systems*, pages 72–86, London, December 1995. Dublin City University, Ireland, Springer-Verlag.

A Ubiquitous Computing Communication and Management Architecture

Markus Lauff

TecO - University of Karlsruhe,
Vincenz Priessnitz Str. 1,
D-76131 Karlsruhe, Germany
markus@teco.edu
http://www.teco.edu/

Abstract. The vision of Ubiquitous Computing requires an infrastructure that is not available yet. Research focussed on Ubicomp applications often is evaluated using proprietary infrastructures. The development of real world Ubicomp infrastructures often fails, because the proposed technologies pay no attention to the integration of existing heterogeneous infrastructures.
The architecture described in this paper focuses on the integration of arbitrary communication protocols, the functional extension of existing resources, and the technology independent application support using a Ubicomp API.
Communication protocols can be integrated using Network Adaptation Modules. Through the use of Device Agents even devices not connected to a network can be integrated or the functions of connected resources can be extended.
The hierarchical management of devices and locations allow a scalable use of the architecture.
The abstraction from real device functions to classes of the same or similar functionality allow users to use different technologies and devices in the well-known fashion.
The Ubicomp API finally allows to develop Ubicomp applications without paying attention on the technologies used to set up the system. Applications can be used in arbitrary environments and will work in a best effort manner.
Keywords: Ubiquitous Computing, Operating System, Networked Appliances

1 Introduction

Ubiquitous Computing is one of the emerging areas in computer science. The vision of ubiquitous computing was introduced by Mark Weiser in his paper "The Computer for the 21st Century" [1] and is based on

- a multitude of appliances in the our environment,
- transparent communication between the appliances,
- a loss of the awareness to operate the appliances as computers.

Another term describing a similar development is the "The Invisible Computer" used by Donald A. Norman' [2]. His approach is to humanize technology, to make it disappear from sight, replaced by a human-centered, activity-based family of information appliances.

Research on this area can be found in the development of wireless communication protocols (e.g. IrDA, BlueTooth), the evaluation of new applications using proprietary technologies [3,4], or the development of architectures supporting communication and management in Ubicomp related environments (e.g. JetSend, Salutation, [5,6]).

Most of these works require new hardware to benefit from their results. It is not possible to integrate existing devices. In contrast to this our approach allows to integrate existing devices and other communication protocols. In our opinion this is essential because there are already many devices that could be used for Ubicomp applications. Furthermore we think that it is wrong to assume that either it is not necessary to use these devices or it is acceptable to replace theses devices with devices using the appropriate technology.

In the original idea of Ubiquitous Computing devices communicate in an ad-hoc fashion. One restriction of ad-hoc communication is that only devices connected to same ad-hoc network technology can participate. The presented architecture uses a Management Service on top of the communication layers to allow network technology independent communication.

Other design goals which led to the presented architecture were

- Location dependent services
  Services can be bound to a location or more general to a set of locations of a specified class (e.g. one service service controlling the room temperature can be used for living rooms whereas another service is used to control the room temperature of bed rooms)
- Support of unreliable, on-/off-line connections
  Todays mobile devices often use communication bound to physical requirements (e.g. infrared communications requires a maximum distance and a small angle to the communication partner). Due to these requirements the connection is often lost – conscious or unconscious – and re-established.
  During the disconnection the device is still existing, and may be used by applications after it is re-attached (e.g. time uncritical messages)
- Technology independent access using an Ubicomp API. The Ubicomp API allows to use services independent from the used technology.

The presented work is structured as follows. The next section "Used Terms" will introduce a few often used terms and their meaning in this paper. After an overview of the architecture, the following sections will go to explain the network adaptation, the addressing and classification of devices, the management of devices and communication, the Ubicomp API and finally a conclusion evaluating the work done and mentioning a few things still to do.

2 Used Terms

2.1 Device / Appliance

We consider all the things we may want to use in our system to be a device. Electrical devices are considered to be appliances and if there is any possibility to remote control the appliance or to communicate with another appliance, it is called a networked appliance.

As you can see in figure 1 the set of networked appliances is a subset of the appliances and appliances are members of the set of devices.

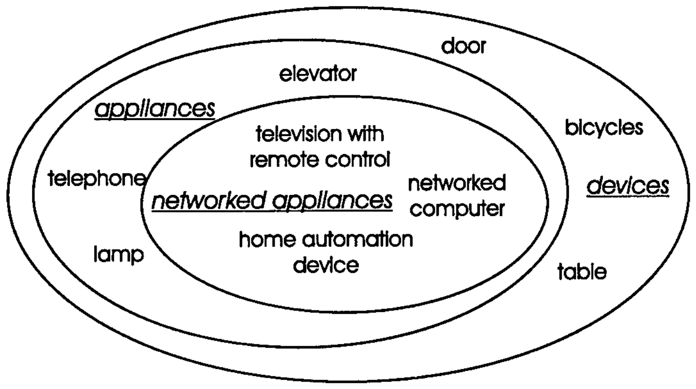


Fig. 1. Devices - Appliances - Networked Appliances

2.2 Environment / Location

A location is an area specified by geographic or symbolic information.

Example 1 (Location).

- room number 1 at the city hall of Karlsruhe
- the city surrounding specified geographic coordinates
- current building

An environment encloses a location and includes all its devices.

2.3 Communication

The term of communication is used whenever an appliance interacts with another appliance to exchange information.

The communication protocol is the way how the communication takes place. In many cases communication protocols are not explicitly named. In theses cases we introduce an own name for this kind of communication.

Example 2 (Communication using Infrared). An infrared remote control operating an appliance uses a communication between the remote control and the device. We name this kind of communication IrPCM (Infrared Pulse Code Modulation).

3 Architecture Overview

The architecture is structured into three layers (Fig. 2).

- Adaptation Layer (UC-AL)
- Service Layer (UC-SL)
- API Layer (UC-API)

The Adaptation Layer and the Service Layer use a modular structure to allow an easy extension.
The communication between the different layers and modules uses TCP/IP as the backbone communication protocol.
The Adaptation Layer allows to add new communication protocols or external systems by adding a corresponding adaptation module.

3.1 Adaptation Layer

The modules of the Adaptation Layer allow to communicate with other communication protocols or systems.
There are two different types of Adaptation Modules defined:

- Network Adaptation
  The Network Adaptation is defines as an application layer communication gateway.
- System Adaptation

The lowest layer, the Adaptation Layer, is working as a communication gateway between different communication protocols and systems.

In contrast to usual network gateways the conversion of the UC-NAL is done with respect to the kind of information transmitted (ISO Reference Model Layer 6 - Presentation Layer).

3.2 Service Layer

This layer consists of the two default modules Management Service and Device Agent Service and can be extended through additional modules.

The Management Service Module is responsible for the management of the devices and environments. Therefore it is organized as a distributed hierarchical directory service to store information about environment, appliances and conditions.

The Device Agent Service allows to extend the functionality of existing networked appliances or even to enable appliances to be a part of the system. A Device Agent is used instead of the actual appliance and acts as a surrogate between the appliance and the system.

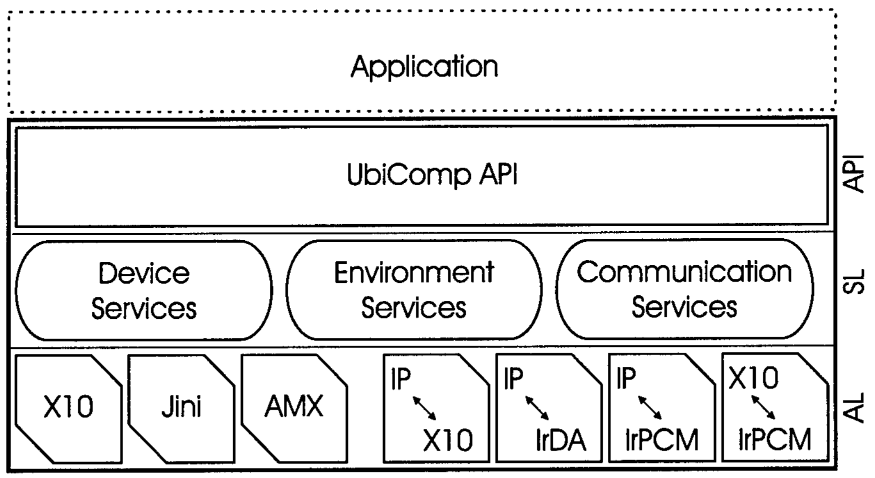


Fig. 2. Architecture Overview

3.3 API Layer

The main function of this layer is – as usual for an API – to hide the internal representation and to allow applications to access the system independent from the actual implementation. The functions offered by the Ubicomp API can be grouped as followed:

- Environment Look-Up
- Environment Control
- Appliance Communication

4 Network Adaptation

The modules located at the Network Adaptation Layer convert information from one communication protocol to another. In contrast to usual network gateways the conversion is done with respect to the kind of information that should be converted. Additionally there is no need to convert everything received. The module can define the set of information accepted for processing. Depending on whether everything is accepted or not the module is named Full-Service (FS) or Partial Service (PS) UC-NAL Module.

The module on the left side of figure 3 is a Partial Service UC-NAL module (indicated by the crossed arrows). This module converts selected information from communication protocol P1 to communication protocol P2.

The module on the right side is a Full Service UC-NAL module. All information received are converted to communication protocol P3.

Example 3 (FS UC-NAL: Serial - IP). A module converting all data received from the serial line to an TCP/IP network is a Full-Service NAL Module.

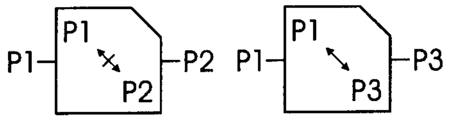


Fig. 3. PS and FS UC-NAL modules

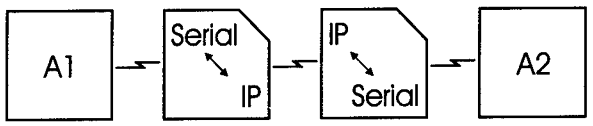


Fig. 4. Network Adaptation - Example 1

Usually this module is used together with a IP-Serial Module (the other direction) to use IP networks as a transit network (Fig. 4)

Example 4 (PS UC-NAL: Serial - X10). The Serial-X10 module shown in figure 5 converts X10 commands received from a serial line to X10 commands using power line. A common Serial-X10 module is a Home Automation Controller receiving commands from a desktop computer to transmit to a X10 device.
The module is realized as a Partial Service because it is not practical to convert all serial information to the slow X10 network. In addition to the X10 commands the module

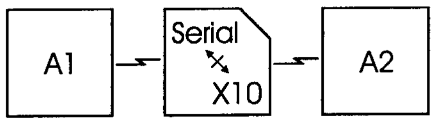


Fig. 5. Network Adaptation - Example 2

might convert IrPCM commands (infrared remote control commands) to transmit over X10 and consequently use the X10 network as a transit network for IrPCM commands.

UC-NAL Module Definition An UC-NAL module is specified by the following properties:

- address (Sec. 5.1)
- source communication protocol
- FS or PS with set of information accepted
- target communication protocol

The address is an unique identifier of the module (Section 7). The system uses this address to enable communication through the module and to locate the module in an environment.

With the help of the above mentioned four properties the network routing (Sec. 7.3) is able to build the routing tables.

5 Addressing / Classification

5.1 Addressing

For the unique identification of devices every device is managed using an address.

To ensure the uniqueness of the address we use an hierarchical naming scheme. The name space can reflect the physical naming or any other naming scheme.

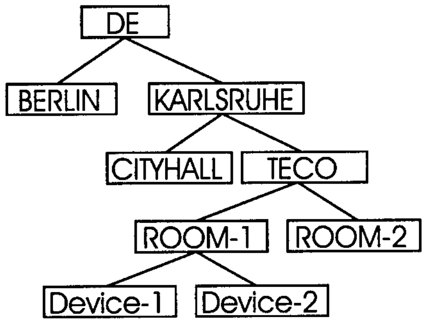


Fig. 6. Hierarchical Addressing

The address of device 2 in room 1 at TecO, Karlsruhe, Germany is
`device-2.room-1.teco.karlsruhe.de`

Besides the hierarchical geographic addressing scheme used in the last example any other unique addressing scheme can be used to identify a device.

other examples:

A car can be addressed using the country code and car number:
`ka-tt-123._carnumber.de`

A person can be addressed using the country code and passport number:
`18728038264995._pp.de`

In cases where more than one address is suitable for a device. One address is chosen as the primary address and the other addresses are declared as aliases (for more detailed information on multiple addresses and routing see also section 7.3).

5.2 Classification

A main advantage of this architecture in comparison to other network architectures is that the application can use devices without caring about functional details of the device.

The device look-up is done on meta information specifying the functions and environment.

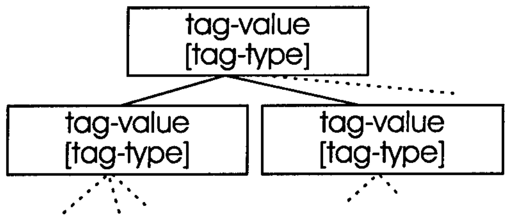


Fig. 7. Classification using Tags

For this purpose the device can use tags to advertise its functions. The use of tags is organized hierarchically (Fig. 7)

The name of the tags, the "tag-type" can be freely chosen. However there is a set of predefined tag types enabling applications to locate and use the devices.

Predefined tag names are: environment, device, agent, dn, kind, cmd, link-agent, link-device.

Example:

Figure 8 shows an example for an ceiling lamp. This lamp (kind: lamp), descriptive name (dn): ceiling lamp, can be controlled using the commands on, off and dim.

The kinds of devices and their commands are not predefined and can be adapted to the systems and users requirements.
Through the use standard values for the kind of devices (e.g. lamp, radio, tv) and standard commands (e.g. on, off, toggle) it is possible to easily control arbitrary devices of the same or similar type.

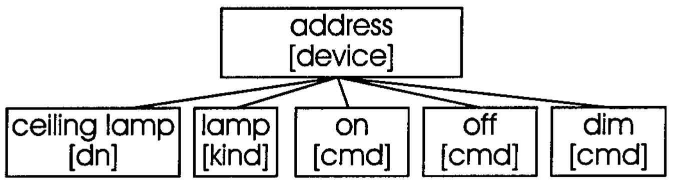


Fig. 8. Classification using Tags - Example

6 Device Agents

A Device Agent is acting for an represented device. Requests sent to the represented device are diverted to the Device Agent.

Depending on the conceptual design of the Device Agent, the Device Agent can completely substitute the device or add more control on it.

A device represented by a Device Agent is flagged with a link-agent tag. The appropriate Device Agent is flagged with a link-device tag (Figs. 9, 11)

The following two examples illustrate typical Device Agent implementations.

Example 5 (Device Agent: Room Door).

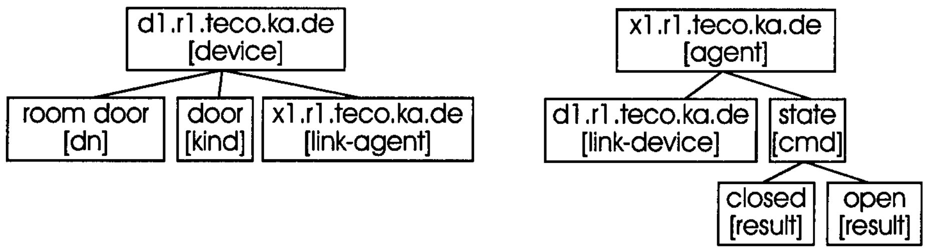


Fig. 9. Device Agent: Room Door

A room door is a device but no appliance. Therefore it cannot communicate. An appliance monitoring the state of the door (sensor) may perform as an Device Agent of the door.

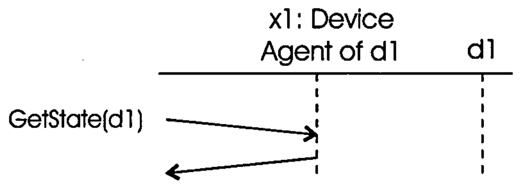


Fig. 10. Device Agent: Room Door - Communication

The request about the state of the door (open, closed) will be redirected to the agent answering on behalf of the door.

Example 6 (Device Agent: TV).

A common remote controllable TV is a networked appliance. A basic function is to switch it on/off. But there is no way to ask the TV if it is currently switched on or off.

A Device Agent of the TV can remember the last used command and answer the question on behalf of the TV.

A Device can be represented by multiple Device Agents using linked chains (Fig. 13).

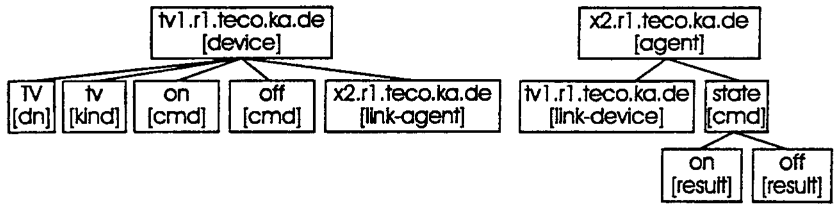


Fig. 11. Device Agent: TV

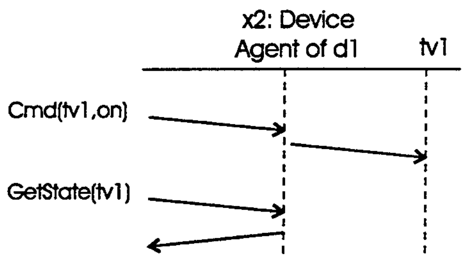


Fig. 12. Device Agent: TV - Communication

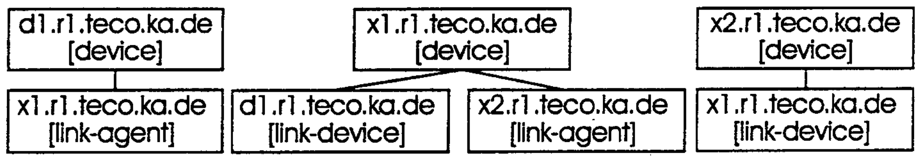


Fig. 13. Chaining Device Agents

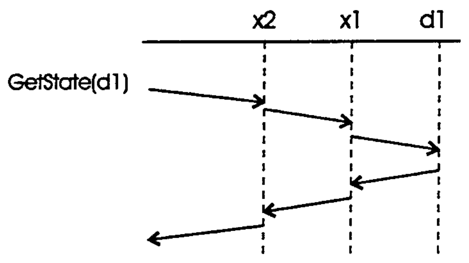


Fig. 14. Chaining Device Agents - Communication

7 Management Service

The Management Service Module is responsible for the management of the devices, environments, and communication. Therefore the Management Service maintains a Directory Management for the devices (section 7.1) and environments (7.2).

The routing tables determined by the Management Service are dynamically built depending on the registered UC-NAL modules, the state of the communication links, and the state of the devices (section 7.3).

7.1 Directory Management

The Directory Management is based on a hierarchical structured scheme.
Each node is defined by a name and a set of attributes (name - value pairs).

Multiple Directory Managements are linked, so that a look-up request of a node not managed by the current Directory Management is forwarded to another instance.

The design of the Directory Management was done in view of well known hierarchical directory services to allow an efficient implementation using for example LDAP.

In practice the Directory Management is maintained by a more intelligent device within a room or building. However it is imaginable that a room contains multiple Directory Managements for security or fault tolerance reasons.

7.2 Environment Management

One major component of the presented architecture is to manage the devices within an environment and to allow applications to use them without knowing them in advance. Therefore the devices are associated as good as possible with the corresponding environment. As already shown in Section 5 the environment is hierarchically managed.

The location model used in this work is based on Ulf Leonhard's PhD thesis "Supporting Location-Awareness in Open Distributed Systems" [Leo98].

The combination of geographic and symbolic location information allows to combine very heterogeneous location information into a usable real world mapping. The heterogeneity of location information is given by the difference of geographic to symbolic location information and on the degree of accuracy.

The next few paragraphs introduce the notation used in this model (for more detailed information see [7]).

The name space for locations addresses to the following issues:

- hierarchical location names for scalability
- the name space can reflect the physical location space be intuitive
- locations have characteristics which are important for location-awareness: Locations can be fixed or mobile, and their extent can be specified by either area geometry or an abstract concept (e.g. "Room")
- hierarchical grouping in path names is used to describe fixed locations. For example, `r1.teco.karlsruhe.de` identifies room number one at the TecO ,Karlsruhe, Germany

- coordinate tuples can be used together with the reference coordinate system. Example: `0-04W.51-3N.0.WGS84` identifies a particular point in the London Area
- abstract locations can use any unambiguous natural language abstractions, e.g. room, river, town or country
- geometric shapes can define a location mathematically described by a geometric structure, such as a circle, a cube, or a rectangle.

The name-space is now constructed using the two dimensions: a geometric area, and the position of this area with respect to some reference point.

$$location ::= \langle area \rangle @ \langle position \rangle \tag{1}$$

$$area ::= \langle locationconcept \rangle \mid \langle geometricdefinition \rangle \tag{2}$$

$$position ::= \langle locatedobject \rangle.\_lo \mid \langle fixedposition \rangle \tag{3}$$

Below are some examples of locations specified with this notation:

`room@r1.teco.ka.de`	Room 1 in TecO in Karlsruhe
`room@Jeff_Magee._lo`	Room where Jeff Magee is currently located
`<5m@Jeff_Magee._lo`	Zone of proximity around Jeff Magee
`<1m@0-04W.51-3N.0.WGS84`	Sphere around a spot somewhere in London

Predicates defined on this name-space are

- Inclusion
- Position
- Distance
- DistanceNamed
- Nearest
- NearestNamed

The presented semi-symbolic model can use geometric and symbolic information and also decouples application representation from the sensor representation of location information. The hybrid model can accept location data in both forms, and all data in the model can be viewed from both perspectives.

7.3 Network Management

The primary information required for communication are managed by the Management Service. At the time of a connection request the Network Management is determining the possible network connections. If there is more than one possibility to establish a network connection, the Network Management chooses the most suitable connection depending on the parameters of the connection request.

As the environments are frequently changing and the routing tables depend on the parameters of the connection request, the tables are defined just in time. Static or even dynamic adapted routing information can not be used, because the connection request can contain arbitrary boundary conditions.

Boundary conditions can for example based on the following properties:

- connection bandwidth/jitter/delay
- reliability of the connection
- fixed/variable costs
- time to establish the connection

8 Ubicomp API

The Ubicomp API Layer allows applications to access the devices without caring about the used technologies. Furthermore it allows applications to abstract from the real functionality of devices and to focus on the necessary and wanted functions.

Example:

A computer that is able to display a TV program on the monitor – after the PC was started, the sound-card initialized, the TV application started and its window maximized – looks for Ubicomp application like a standard TV. It is switched on/off the channel is selectable and volume can be tuned. All these things without keeping in mind which kind of TV is used.

The functions of the Ubicomp API can be grouped to the following:

- location/environment discovery
- device discovery and control
- communication requests and connections
- security/access control and configuration
- system control and configuration

9 Conclusions

This work presents an Operating System Support for Ubicomp Applications. It is based on an architecture allowing the flexible integration of heterogeneous devices, communication protocols or even of different automation standards.

The different communication protocols are integrated using UC-NAL module for adaptation.

The maintenance and administration of location information, the devices in the environment, and the communication are done by the Management Service. The second module of the Service Layer, the Device Agent Service, allows to extend and adapt the functions and behavior of the devices.

Applications using this architecture can access the necessary devices through the Ubicomp API without handling special communication protocols, investigating the dynamic changes in the environment, or the users preferences.

A prototypical implementation was done using a X10 Home Automation System, a infrared remote control gateway, a parallel relay card, a LDAP server on UNIX, and Personal Digital Assistants (PDAs) to access the system using IrDA.

With this implementation many conceptual design issues have been evaluated and proven. Further work considers the security aspects for user management and access control [8]. Another area for further developments is the design of Ubicomp application using new concepts of user interaction [9].

References

1. Mark Weiser. The computer for the twenty-first century. *Scientific American*, pages 94–104, 1991.
2. Donald A. Norman. *The Invisible Computer: Why Good Products Can Fail, the Personal Computer Is So Complex, and Information Appliances Are the Solution.* The MIT Press, 1998.
3. Mark Weiser. Open house. Interactive telecommunications program, New York University, March 1996.
4. Jeremy R. Cooperstock. From the flashing 12:00 to a usable machine: Applying ubicomp to the vcr. In *CHI97 Proceedings*, March 1997.
5. Markus Lauff. NA3 - networked appliances automation architecture. In *Proceedings of the IEEE Workshop on Networked Appliances (IWNA)*, Kyoto, Japan, November 1998.
6. T. Drashansky, S. Weerawarana amd A. Joshi, R. Weerasinghe, and E. Houstis. Software architecture of ubiquitous scientific computing environments for mobile platforms. *Mobile Networks and Applications*, 1(4):421–432, 1996.
7. Ulf Leonhard. *Supporting Location-Awareness in Open Distributed Systems.* PhD thesis, University of London, May 1998.
8. Yossi Tsuria. Security considerations in networked appliances. In *Proceedings of the IEEE Workshop on Networked Appliances (IWNA)*, Kyoto, Japan, November 1998.
9. Michael Beigl, Albrecht Schmidt, Markus Lauff, and Hans W. Gellersen. Ubicomp browser. In *4. ERCIM Workshop on User Interfaces for All*, October 1998.

Meeting the Computational Needs of Intelligent Environments: The Metaglue System

Michael H. Coen, Brenton Phillips, Nimrod Warshawsky, Luke Weisman, Stephen Peters, and Peter Finin.

MIT Artificial Intelligence Lab
545 Technology Square
Cambridge, MA 02139
mhcoen@ai.mit.edu

Abstract. Intelligent Environments (IEs) have specific computational properties that generally distinguish them from other computational systems. They have large numbers of hardware and software components that need to be interconnected. Their infrastructures tend to be highly distributed, reflecting both the distributed nature of the real world and the IEs' need for large amounts of computational power. They also tend to be highly dynamic and require reconfiguration and resource management on the fly as their components and inhabitants change, and as they adjust their operation to suit the learned preferences of their users. Because IEs generally have multimodal interfaces, they also usually have high degrees of parallelism for resolving multiple, simultaneous events. Finally, debugging IEs present unique challenges to their creators, not only because of their distributed parallelism, but also because of the difficulty of pinning down their "state" in a formal computational sense. This paper describes Metaglue, an extension to the Java programming language for building software agent systems for controlling Intelligent Environments that has been specifically designed to address these needs. Metaglue has been developed as part of the MIT Artificial Intelligence Lab's Intelligent Room Project, which has spent the past four years designing Intelligent Environments for research in Human-Computer Interaction.

Introduction

Research on highly interactive spaces, generally known as Intelligent Environments, has become quite popular recently. Although their precise applications, perceptual technologies, and control architectures vary a great deal from project to project, the *raisons d'être* of these systems are generally quite similar. They are aimed at allowing computational systems to understand people on our own terms, frequently while we are busy with activities that have never before involved computation. IEs

This material is based upon work supported by the Advanced Research Projects Agency of the Department of Defense under contract number F30602—94—C—0204, monitored through Rome Laboratory.

seek to connect computational systems to the real world around them and the people who inhabit it.

This paper presents what we believe are general computational properties and requirements for IEs, based on our experience over the past four years with the Intelligent Room Project at the MIT Artificial Intelligence Lab. Although many of the published descriptions of IEs [4] differ in their particulars, it is clear that we have not been alone in confronting some of the frustrating aspects of engineering these complex systems.

Based on this experience, we have developed Metaglue, a specialized language for building systems of interactive, distributed computations, which are at the heart of so many IEs. Metaglue, an extension to the Java programming language, provides linguistic primitives that address the specific computational requirements of intelligent environments. These include the need to: interconnect and manage large numbers of disparate hardware and software components; control assemblies of interacting software agents *en masse*; operate in real-time; dynamically add and subtract components to a running system without interrupting its operation; change/upgrade components without taking down the system; control allocation of resources; and provide a means to capture persistent state information.

Metaglue is necessary because traditional programming languages (such as C, Java, and Lisp) do not provide support for coping with these issues. There are currently several other research systems for creating assemblies of software agents [7,8,9], which provide low-level functionality, e.g., support for mobile agents and directory services. These features are necessary but not sufficient. Because Metaglue provides high-level tools specifically relevant to creating software controllers for IEs, we hope to make it available for more widespread use by the IE community.

Much of our discussion will focus on *Hal*, our most recently constructed Intelligent Room [3,5], where approximately 100 Metaglue agents control Hal and interconnect its components. However, we believe the issues raised here extend beyond the particulars of Hal and are important for a wide range of intelligent environments.

Hal is a small room within our lab and is equipped with microphones, seven video cameras, and a variety of audio-visual output devices that it can directly control. Hal was designed to explore a wide range of interactions involving futuristic residential spaces – stressing quality of life – and commercial spaces – stressing information management. We have therefore created applications in Hal that support entertainment, teleconferencing, business meetings, military command post scenarios, and information retrieval.

Next, we expand on our list of computational properties for IEs and examine the reasons behind them. We then discuss design considerations of the Metaglue system, and how it specifically addresses the perceived needs of IEs. In this, we directly trace how the issues raised in the next section are satisfied by capabilities incorporated into Metaglue. Finally, we close with an evaluation of the Metaglue system and directions for future research.

Computational Properties of Intelligent Environments

Intelligent Environments by and large share a number of computational properties due to commonalties in how they internally function and externally interact with their users. Of course, not every IE will identically share all of these characteristics, but we believe examining them even briefly makes concrete many of the issues that IE designers are faced with and should address directly. We not only hope to further discussion on these issues in the IE community, but to motivate the development of other general purpose tools such as Metaglue.

We note that when multiple people are allowed to interact simultaneously with a single IE, many of the issues discussed below are greatly exacerbated. Space limitations preclude addressing this issue in detail.

Distributed, modular systems need computational glue

Intelligent Environments contain a multitude of subsystems comprising their perceptual interfaces, software applications, hardware device connections, and mechanisms for internal control. Even though each IE is created in its own way for its own purpose, IEs are generally built out of similar components.

Thus, IEs require some way to *glue* all of these components together and coordinate their interactions. These components also generally cannot co-exist on a single computer, due to hardware constraints and the need for environments to respond in real-time to their users. It is not uncommon for individual computer vision or speech understanding applications to consume the resources of an entire workstation, and there is no reason to believe this exclusivity will be diminished as processor speeds increase in the future. Many of these systems perform progressive real-time searches that naturally generalize to consume all increases in available computational power.

Frequently, these components, either off the shelf or research programs in their own right, are not designed to work together, so not only must they be connected, but there needs to be some way of expressing the "logic" of this interconnection. In other words, inter-component connections are not merely protocols, but must also contain the explicit knowledge of how to use these protocols. Thus, viewing the connections simply as Application Programming Interfaces (APIs) is insufficient. For example, consider connecting a speech recognition system and a web browser, so that users can navigate links by speaking the text contained in them. Here, the computational *glue* would include a mechanism that dynamically updates a recognition grammar with the link text whenever the user goes to a new web page; simply having APIs to both of these applications is necessary but not sufficient.

More generally, enormous amounts of control code go into building IEs, much of it dealing with how connections among its pieces should be managed. (See [1,2].)

Resource management is essential

Interactions among system components in an IE can be exceedingly complex. Resources, such as video displays or computational power, can be scarce and need to be shared among different applications. For example, in Hal, multiple computer vision systems share individual video cameras because they are a relatively expensive resource [6]. Conflicts can occur when multiple applications in an IE all want to display information on a single video display or speak to the user simultaneously. One of the largest surprises while developing Hal was that even seemingly simple issues, such as displaying a video, have wide spread repercussions on other parts of the running system because their resources are pulled out from under them. We discuss this at greater length in the next section. Finally, even in an environment with ample resources for a single user, conflicts can unknowingly arise when multiple people attempt to interact with it simultaneously.

Thus, IEs need sophisticated resource management capabilities, particularly to let them scale properly as new applications and capabilities are added.

Configurations change dynamically

IEs can be highly dynamic systems. In the prescient words of Weiser – referring to Ubiquitous Computing but equally relevant to IEs – *"New software ... may be needed at any time, and you'll never be able to shut off everything in the room at once ... functionality may shrink and grow to fit dynamically changing needs"* [12]. People may come and go at will, bringing with them devices such as PDAs that temporarily connect with an IEs existing computational infrastructure.

In a developing system such as Hal, new hardware and software components are incorporated on a regular basis. It should be possible to add them to a running system without restarting unrelated components. In fact, under many circumstances, new permanent components should dynamically integrate themselves into an IE without interrupting its operation at all.

Even within the confines of a static IE, users may readily switch between different aspects of its functionality. For example Hal supports teleconferencing and information management applications and users readily switching between the two is quite natural; often during meetings the need arises to get more information about something. These "context" changes can have far reaching effects. For example, an IE may need to simultaneously start new underlying applications, activate different speech recognition grammars, and modify the configurations of other perceptual systems.

State is precious

Not only may new components need to be incorporated into a running IE, but pieces of a running system may need to be reloaded into it as well. As with any other software engineering effort, creating IEs require an iterative edit-recompile-run process while testing new features and eliminating bugs. However, if the entire system had to be restarted each time one of its components changed, development would be prohibitively time-consuming. Our Hal environment has literally dozens of hardware and software components connected to approximately 100 Metaglue agents. Having to bring this system completely down to modify it would long ago have made further development a cause of endless frustration.

Furthermore, IEs acquire state through interactions with users. Attempting to trace a bug by forcing a person to repeat a sequence of interactions that may have spanned several hours would be outrageous. To exacerbate the problem, state is acquired not only through human interactions, but also through any activities Hal has engaged in, such as information retrieval. A weather report or CNN headline that caused Hal to take some action may have long since changed and is not recoverable.

The most critical part of Hal's state comes from information it learns while observing its users. Hal has several machine learning systems for learning about users' preferences and activities. These systems have no straightforward way to unlearn and return to a previous coherent state. Checkpointing in the style of reliable transaction systems can partially ameliorate these problems with respect to the local state of individual components. However, when IEs are asynchronous and distributed, repeating a particular *global* state can be, practically speaking, impossible to achieve. (One technique we have been investigating is allowing an IE to essentially simulate itself by replaying previously observed and recorded events.)

Thus, there is a clear need for an IE's software architecture to permit a kind of dynamism rare in conventional computational systems. We would like to stop, modify, and reload components of a running system and have them reintegrate into the overall computation with as much of their state intact as possible.

IEs model the parallelism of the real world

Supporting natural human computer interaction requires that IEs have some handle on multiple ways that a user may interact with them. People speak, gesture, move, and emote simultaneously, and IEs need to have some capacity to cope with this, even on the part of a single user. This is not to say they need to understand the full range of human discourse to be useful. Far from it, IEs that consistently – and most importantly, predictably – understand a small subset of interactions are far preferable from an HCI perspective to ones that always leave users guessing if some particular input will be understood.

Nevertheless, IEs generally have multimodal interfaces which requires they have sufficient parallelism for resolving multiple, simultaneous events. For example, if a user walks to a displayed map, points somewhere and says, *"What's the weather here today?"* and then immediately walks away, the system must be able to discover where they were pointing when they said the word *"here."* Dealing with multiple users simultaneously again simply exacerbates the problem.

Thus, IEs need at least as much parallelism as the phenomena they are trying to understand. In fact they may need a great deal more for background processing, which leads to the next item.

Real-time response

It almost goes without saying that IEs need to be responsive to their users. Particularly due to the fact that many IEs do not have traditional computer monitors so users can get a handle of the system's inner activity, it is astonishingly frustrating when an IE does not respond quickly to user input. This is another point supporting the basic need for parallel architectures in an IE. The parts of the system that acknowledge and react to users must be immediately responsive even if other parts of the system, for example, in the midst of processing an information retrieval query, require more time to respond.

This also requires that the mechanism for interconnecting an IE's components and processing their data be able to keep up with the underlying external systems. For example, in Hal, five C-language-based computer vision systems, each producing several hundred dimensional data vectors at a rate of up to 30 a second, all connect to a Metaglue-based visual event classification system which must process all this data in real-time.

Debugging is difficult

Independently of IEs, debugging distributed, asynchronous systems can be a nightmare. If some high-level system event fails to occur, determining which component is to blame is usually a long, involved process. Furthermore, understanding the operation of distributed, loosely coupled components running in parallel – as does the controller for an IE – where different serializations can have different system-wide effects, is best, but rarely successfully, avoided. In an IE, this problem is made all the worse by the presence of many, sometimes exotic, hardware components, such as video multiplexers, that themselves have internal state that may only be imperfectly modeled in their software drivers.

Good software engineering practices go a long way towards dealing with this problem, but a more comprehensive solution would require the development of new types of debugging strategies. (In the next section, we see that Metaglue only makes the most preliminary efforts in this direction.)

Metaglue

We first discuss the design of Metaglue from a programming language perspective, to give potential users a sense of what it would be like to work with it. We then proceed to illustrate how particular features in Metaglue address many of the computational requirements for IEs discussed in the previous section. By necessity, this section is intended only to sketch and motivate the capabilities of the Metaglue system and should not be viewed as a complete description of the language. More detail about Metaglue's internals can be found in [10,11]. (Some examples in this section require cursory knowledge of the Java programming language.)

The Design

Metaglue is an extension to the Java programming language that introduces a new *Agent* class. By extending this class, user-written agents can access the special Metaglue methods discussed below. Metaglue has a post-compiler, which is run over Java-compiled class files to generate new Metaglue agents. Metaglue also includes a runtime platform, called the *Metaglue Virtual Machine*, on which its agents run. The overhead added by this infrastructure to standard Java programs that are turned into Metaglue agents is negligible.

Our goal with Metaglue was to add a very small number of primitives to the Java language to make it easy to write agents. Method invocations between agents, even if they are on different workstations, look exactly like local method calls in Java. Thus, Metaglue agents, minus the few Metaglue-specific primitives, look almost exactly like ordinary Java programs. This makes it easy to transform regular Java source files into Metaglue agents, which enormously adds to Metaglue's value as computational glue. We call the process of transforming previously existing programs into agents *wrapping*.

Almost as much time has gone into formulating Metaglue's semantics as to programming the system itself. We sought to provide a focused set of primitives for managing systems of distributed, interacting agents and to avoid the temptation of creeping featurism. By stressing simplicity, anyone proficient in Java can pick up Metaglue very quickly, and the small number of new primitives makes it easy to learn, remember, and use the system.

In the remainder of this discussion, it will be helpful to keep in mind that running a Metaglue system first involves starting Metaglue Virtual Machines on all the computers that are involved. Our machines are generally configured to start these when they are booted.[1]

The Capabilities

Metaglue offers the following capabilities, each of which we will address in turn:

1. Configuration management
2. Establish *and maintain* the configuration each agent specifies
3. Establish communication channels between agents
4. Maintain agent state
5. Introduce and modify agents in a running system
6. Manage shared resources
7. Event broadcasting
8. Support for debugging

Metaglue has a powerful naming scheme for agents that is beyond the scope of this document. We will use here the simplest form of it, the name of the Interface file of an agent, which is in the Java class package format, e.g., an agent for controlling a television might be referenced by *device.Television.*

1. Configuration Management

Metaglue has an internal SQL database for managing information about agent's modifiable parameters (called Attributes), storing their internal persistent state, and giving agents fast, powerful database access.[2]

Attributes contain information that might otherwise be hardcoded inside agents and difficult to modify, for example, what workstation the agent needs to run on or parameters that affect its operation. Metaglue has a web-based interface for modifying Attributes, which can be changed even while an agent is running. This is one of Metaglue's mechanisms for both configuring a system of agents and interacting with it while it is operational.

[1] For reference, this has the computational overhead of running one Java Virtual Machine, which is close to unnoticeable on modern Pentium-based systems.

[2] The use of an internal database helps enormously in dealing with Java's poor file access capabilities. Agents use the database rather than store information in files, which is particularly important because agents can move to different machines while they are running.

This is code an agent would use to get its *location* Attribute from Metaglue's built in database:

```
Attribute location = new Attribute("location");
System.out.println("I run on " + location.getValue());
```

2. Agent Configurations
Metaglue agents can specify particular requirements that the system must insure are satisfied before they are willing to run. These can include the name of a particular computer they must be run on; specifications for particular types of hardware they require access to; and more abstract capabilities that must be available on whichever Metaglue Virtual Machine (MVM) they are run on. These are expressed with the `tiedTo()` primitive, as in:

```
tiedTo(location.getValue());
tiedTo(capability.FrameGrabber);
tiedTo(device.Television);
```

If an agent is started on an MVM that does not meet its stated requirements, Metaglue will move it somewhere else that does. If Metaglue needs to restart an agent due to localized hardware or software failure, it will attempt to find an alternative MVM on which to run the agent that also satisfies these requirements. (See item 5.)

3. Agent Connections
Because Metaglue is intended as computational glue, it needs to establish paths of communication between agents, regardless of where they are running. The `reliesOn()` primitive connects agent with capabilities they can request services from. For example, to use Hal's speech synthesizer, an agent might contain:

```
Agent speechSynthesizer = reliesOn(speech.Synthesizer);
speechSynthesizer.say("Hello! How are you?");
```

The reference to `speech.Synthesizer` refers to an abstract capability, not a particular agent. Because agents refer to each other by their capabilities and not directly by name, new agents can easily be added to the system that implement preexisting capabilities without modifying any of the agents that will make use of them. (A more sophisticated way of obtaining capabilities is described in the Metaglue resource manager below.)

Metaglue will try to locate an agent that provides the requested capability on any of the system's computers' MVMs and return a reference to it to the caller. Metaglue has an internal directory called a *Catalog* that it uses to find agents once they are started. Metaglue agents automatically register their capabilities with the Catalog when they are run.

If no agent offering this capability is found, Metaglue automatically starts one and invisibly insure that it continues running as long as it is in use.

The `reliesOn()` primitive makes it very easy to interconnect agents with a single line of code. Reliance is also a transitive operation. For example, starting the single Hal *demo* agent results in the entire Hal system being loaded because of their chain of reliances. Also, because they have been formally relied upon, Metaglue will attempt to insure that they continue running indefinitely, as discussed below.

4. Agent State

Agents can use Metaglue's `freeze()` and `defrost()` primitives to store and subsequently retrieve their fields from Metaglue's internal SQL database. The standard way of doing this is having an agent directly freeze its state when it is shutting down (or at any other appropriate time), and subsequently defrost itself in its constructor the next time it is started up. Other aspects of an agent's state, e.g., its connections to other agents, are internally maintained by Metaglue and generally do not need to be specifically managed by the agents themselves.

As of yet, we do not have a well-defined schema for capturing the global state of all the agents in a running Metaglue system.

5. Modifying a running system

Metaglue will attempt to keep a running system of agents alive. If an agent is manually stopped, for example, during debugging, the agents that rely upon it will by default simply wait for it to return in the event they need to access it. When the user restarts the agent, it will reload its frozen state and simply pick up where it left off, first dealing with any pending requests from other agents.

It is also possible to programmatically specify actions, other than simply waiting, that an agent can do if someone it relies upon is stopped. For example, it might temporarily switch to another active agent that offers the same capabilities. Metaglue's resource management can help the system in the event capabilities must be shared due to part of the system being unavailable.

If an agent dies because of unanticipated hardware or software failure, Metaglue will try to restart it automatically, switching to another MVM if necessary, but still meeting the agent's required configuration if possible. It is important to note that unanticipated crashes may cause state information to be lost, and agents who are sensitive to this should refuse to be automatically restarted. For example, an agent that controls Hal's lighting systems may not know whether the lights are on or off if it is restarted after a crash because the state information it defrosts may be inaccurate. However, in that case, it can simply ask Hal's vision agents whether they can see anything, and thereby determine the state of the lights in the room. An agent running part of an application, however, could be started out of sync with the rest of the system, and manual intervention may be required to correct the problem.

Interestingly, the Metaglue system is itself recursively constructed out of a special set of Metaglue agents. These agents have the full functionality of the system available to them, so they can for example, use Hal's speech synthesizer to let users know of internal problems in the system and ask for help resolving them.

6. Managing shared resources

Among the largest and most complex systems in Metaglue is its resource manager. Before it existed, agents in Hal simply grabbed the resources they needed and configured them at will. That a resource management system was necessary became apparent when Hal developed to the point that its multiple applications conflicted with one another and could no longer be run simultaneously. Additionally, for an agent to simply rely on the resources it wants to use, it has to know both what resources exist and are available. As devices and other agents dynamically come and go in the system, this means that every agent would need to keep track of the different resources that offer the sets of capabilities it needs. We discussed above that agents may temporarily make use of substitutes if the agents they generally rely upon are unavailable. Where should that knowledge of possible alternatives come from?

The resource system in Metaglue allows agents to request functionality at a very high-level, without being concerned with how it is provided or resolving resource conflicts among themselves. Metaglue has a hierarchical set of *dealer* agents that are responsible for distributing resources to the rest of the system. There are a wide assortment of different prototype dealer agents available, each of which has its own specified internal logic for performing allocation, substitution, etc. These dealers can be used directly by Metaglue programmers or extended to customize their operation.

Dealers not only give out resources, but they can withdraw previously allocated ones to redistribute them, based on any of several priority and fairness schemes. For example, there are dealers in Hal for allocating televisions, video projectors, and displays in general. An agent must use the dealers to gain access to any of these. If a higher priority agent needs access to a particular display, it will be temporarily withdrawn from the agent who has allocated it until it becomes available again, at which point it will be given back.

7. Event Broadcasting

In addition to agents making direct requests of one another through method calls, Metaglue agents can pass messages among themselves. Agents can register with other agents, including the Metaglue system agents, to find out about events going on in the system. For example, an agent in Hal interested in greeting people by name when they walk inside the room, simply registers with the vision-based Entry agent to request messages about *entrance* events where the identity of the person can be determined. When these events occur, it receives a message and uses the agent offering speech synthesis capability to say hello to them.

We also use event broadcasts to notify groups of agents about context shifts in room applications to dynamically and uniformly modify Hal's behavior.

8. Debugging

Metaglue has a graphical interface for examining a running system of agents called the Catalog monitor. It displays all running agents and their reliance interconnections. Clicking on an agent brings up a window in a read-eval-print loop, in which users can interactively call the agent's methods.

Metaglue also has a logging facility to manage and centralize agents' textual output. This can be useful for programmers to watch the output of particular agents without worrying about where they are running or where their output streams are being printed.

We have found these capabilities quite useful, but would still prefer source level debugging of remote agents, a dynamic object browser, and ways to set breakpoints over whole groups of agents simultaneously. At least some of these capabilities promise to be available shortly in commercial Java products and we hope to make use of them during Hal's continued development.

Discussion

Evaluating the merits of a programming language can defy objectivity. Nonetheless, Metaglue has been extraordinarily useful in building Hal, and it is highly doubtful Hal would have reached its present level of development with it. Metaglue is a very stable system, and we have left large assemblies of agents running for up to a week without any difficulties. (These systems were eventually stopped for development purposes.)

We now reexamine each of the previously mentioned properties of IEs in the context of Metaglue.

Distributed, modular systems need computational glue
Metaglue not only provides a channel to interconnect Hal's components, but it also provides the means to build applications for Hal. Rather than use a special communication mechanism, such as CORBA or KQML, separate from the system's internal controller, Metaglue allows us to reduce the amount of infrastructure by providing for both communication and control with a much lighter-weight system.

Resource management is essential
The resource management system in Metaglue not only offers a wide range of default behaviors, but it is easily customizable through Java's class extension mechanism. It is among Metaglue's most developed systems and we are in the process of incorporating it into the applications that predated it.

Configurations change dynamically
Metaglue offers several mechanisms for coping with dynamically changing systems. The Configuration Manager and Attribute system allows users to reconfigure agents while they are running. The fact that agents refer to each other by abstract capabilities means that new agents can be incorporated into a running system without modifying any of the agents that might rely upon them. Metaglue's ability to start and stop agents while leaving the rest of the system running allows us to dynamically "hotswap" components of a running computation. Finally, by substituting new resource managers into a running system, new functionality can be added that previously no agents were aware of.

State is precious
Metaglue offers support for persistent local state in agents via its freeze and defrost mechanisms. Notions of global state, however, remain illusive concepts.

IEs model the parallelism of the real world

Java is inherently multithreaded, which Metaglue inherits from it. Metaglue's resource management allows agents running in parallel to avoid conflicting with one another. The event broadcast mechanism also simplifies communication among interacting groups of software agents running simultaneously.

Real-time response

The amount of overhead Metaglue adds to Java is minimal. Our avoidance of heavyweight, specialized communication packages allows Metaglue agents to essentially run as quickly as Java's Remote Method Invocation system. Metaglue is now incorporated into "tight" loops in our code, along the most processor intensive critical paths, such as in our computer vision systems. The development of JIT compilers for Java has enormously reduced our need to place perceptory components of our system into external C-language libraries.

Debugging is difficult

Metaglue certainly makes it possible to debug distributed agents systems, but one can hope for more. There is reason to believe the Java community as a whole shares some of this interest and will takes steps in this direction.

Future directions

We are presently incorporating an expert system into Metaglue to allow more sophisticated reasoning about system configuration and resource management. We are also creating a machine learning extension to Metaglue, which will incorporate pieces of the system described in [6].

References

1. Bobick, A.; Intille, S.; Davis, J.; Baird, F.; Pinhanez, C.; Campbell, L.; Ivanov, Y.; Schütte, A.; and Wilson, A. Design Decisions for Interactive Environments: Evaluating the KidsRoom. *Proceedings of the 1998 AAAI Spring Symposium on Intelligent Environments.* AAAI TR SS-98-02. 1998.
2. Coen, M. Building Brains for Rooms: Designing Distributed Software Agents. In Proceedings of the Ninth Conference on Innovative Applications of Artificial Intelligence. (IAAI97). Providence, R.I. 1997.
3. Coen, M. Design Principles for Intelligent Environments. In Proceedings of The Fifteenth National Conference on Artificial Intelligence. (AAAI98). Madison, Wisconsin. 1998.
4. Coen, M. (ed.) Proceedings of the 1998 AAAI Spring Symposium on Intelligent Environments. AAAI TR SS-98-02. 1998.
5. Coen, M. The Future Of Human-Computer Interaction or How I learned to stop worrying and love My Intelligent Room. IEEE Intelligent Systems. March/April. 1999.
6. Coen, M., and Wilson, K. Learning Spatial Event Models from Multiple-Camera Perspectives in an Intelligent Room. *In submission.*
7. General Magic. *Odyssey (Beta 2)* Agent System Documentation. http://www.genmagic.com/agents.
8. Lange, D. and Oshima, M. Programming and Deploying Java Mobile Agents with Aglets. Addison Wesley. 1999.
9. ObjectSpace, Inc. *ObjectSpace Voyager Core Package Technical Overview (Version 1.0).* December 1997. http://www.objectspace.com/voyager/whitepapers.
10. Phillips, B. Metaglue: A Programming Language for Multi-Agent Systems. M.Eng. Thesis. Massachusetts Institute of Technology. 1999.
11. Warshawsky, N. Extending the Metaglue Multi-Agent System. M.Eng. Thesis. Massachusetts Institute of Technology. 1999.
12. Weiser, M. The Computer for the 21st Century. *Scientific American.* pp. 94-10, September 1991

LEARNING AND INTERPRETATION

Learning Spatial Event Models from Multiple-Camera Perspectives

Michael H. Coen and Kevin W. Wilson

MIT Artificial Intelligence Lab
545 Technology Square
Cambridge, MA 02139
{mhcoen, kwilson}@ai.mit.edu

Abstract. Intelligent environments promise to drastically change our everyday lives by connecting computation to the ordinary, human-level events happening in the real world. This paper describes a new model for tracking people in an intelligent room through a multi-camera vision system that learns to combine event predictions from multiple video streams. The system is intended to locate and track people in the room, determine their postures, and obtain images of their faces and upper bodies suitable for use during teleconferencing. This paper describes the design and architecture of the vision system and its use in Hal, our most recently constructed intelligent room.

1 Introduction

For the past four years, the MIT Artificial Intelligence (AI) Laboratory has been developing platforms for research in Human-Computer Interaction (HCI) as part of the Intelligent Room Project. This work has been motivated by the following simple observation: computers today are generally used for things that are computational, such as reading e-mail, and for most of us, the majority of our lives are spent doing non-computational things, such as taking baths and eating dinner. Historically, however, computational systems have had no connection with the ordinary, human-level events going on in the world around them. Thus, they have no way to participate in the everyday lives of the people who use them.

In the Intelligent Room Project, we are interested in creating spaces – generally known as Intelligent Environments (IEs) – in which computation is seamlessly used to enhance ordinary, everyday activities. We want to incorporate computers into the real world by embedding them in regular environments, such as homes and offices, and allow people to interact with them the way they do with other people. The user interfaces of these systems are not menus, mice, and keyboards, but instead gesture, speech, affect, context, and movement. Their applications are not word processors and spreadsheets, but smart homes and personal assistants. Instead of making

This material is based upon work supported by the Advanced Research Projects Agency of the Department of Defense under contract number F30602—94—C—0204, monitored through Rome Laboratory.

computer interfaces for people, we believe it is of more fundamental value to make people interfaces for computers.

Fortunately, over the past few years, developments in the field of computer vision have begun to provide many opportunities for exploring new types of HCI. These allow computational systems to, among other things, determine people's identity, physical location, gaze direction, facial expression, hand gestures, and activities (for example, see [8,13]). However, the requirements placed on machine vision systems used for IEs are significantly different from the requirements placed on machine visions systems used in laboratory or industrial settings. A vision system that works beautifully under structured, carefully orchestrated laboratory conditions may be of little use in environment where people are encouraged to interact naturally, without worrying about which direction they are facing or even whether they stay in the field of view of some camera.

This paper describes a new model for tracking people in an IE in the context of *Hal*, our most recently constructed intelligent room. In this model, locations in a room that are likely to contain people are used as reference points to interactively train a multi-camera vision system. It learns to combine event predictions from multiple video streams in order to locate people at these reference points in the future. The system can also dynamically track people as they move between these locations. Because most rooms have natural attractors for human activity, such as doorways, furniture, and displays, the selection of training points is generally readily apparent from the layout of the room.

Our interest is how to make it easy to set up and maintain this type of vision system in an environment such as Hal, which currently contains four fixed and three computer-steerable cameras. The vision system described here is intended to locate and track people in the room, determine their postures, and obtain images of their faces and upper bodies suitable for teleconferencing.

Although it may seem extravagant to outfit a small room with so many cameras, the increasingly low-cost of hardware for machine vision makes it quite feasible to do so. However, installing a large number of vision systems in a single room raises the following questions, which the rest of this paper addresses:

1. How can the vision systems learn the types of events they can see?
2. How can individual cameras be shared by multiple vision systems?
3. How can we learn the correspondences among events in the fields of view of different cameras?
4. How can the vision systems learn how their visual fields overlap so they can reinforce each other through communication?
5. How can we avoid the need to precisely calibrate the cameras?

We argue in this paper for a particular approach that has been valuable in the development of Hal, which has been equipped with computer vision systems specifically designed for coping with the difficulties expected to arise during unstructured interactions. The primary components of this system's architecture are its generalizability, redundancy, and ability to learn spatial events without requiring elaborate training scenarios.

We first discuss design level issues for the system, including why we used computer vision as opposed to other possible technologies. Next, we describe Hal's layout, components, and the types of applications intended to run within it. Finally, we discuss the architecture of the vision systems and how well this system has satisfied our design goals.

2 Design Considerations

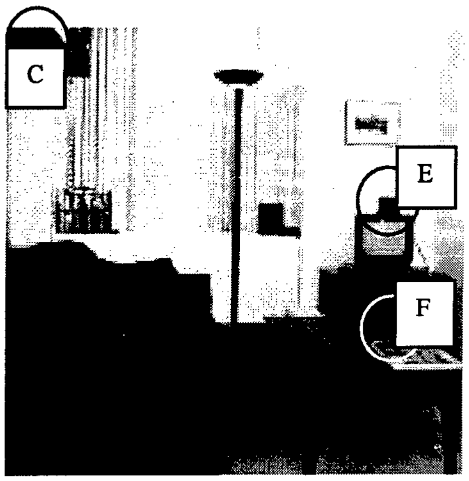
Research in Intelligent Environments has been heavily influenced by the work done in Xerox PARC's Ubiquitous Computing (UbiComp) Project [14,15]. UbiComp has avoided AI approaches to HCI for many of the perceptory tasks at which computer vision excels. Instead, hardware sensors – more reliable in the early 1990s but highly limited in their perceptory capabilities – were employed to gather information about people interacting with UbiComp systems. For example, someone could be tracked in a building by wearing an infrared-transmitting active badge. Similarly, a room could determine that a person was sitting in a chair via a pressure switch in the seat cushion.

However, machine vision techniques can do a far better job of allowing IEs to understand the activities going on within them than can simple sensing technologies. For example, although a pressure sensor in a chair may be able to register that someone has sat down, it is unlikely to provide any other information about her, such as who she is or which way she is looking; active badges or motion detectors [9] can indicate that someone has entered a room, but they can not, for example, determine what objects she is pointing at while speaking. Computer vision systems can provide much richer descriptions of human activities, which are essential for supporting natural interactions.

Non-vision technologies are also highly encumbered – furniture must be constructed or retrofitted with sensors and people must clip devices onto their bodies for a UbiComp system to be aware of them. Vision systems sense at a distance, so that people and furniture do not need to any special augmentation for an IE to perceive them. They also have far greater flexibility, in that once the video hardware is installed, different image processing algorithms can be applied to the video streams without requiring changes to the system's physical infrastructure. Finally, their video streams are reusable by other systems, such as in teleconferencing or surveillance applications.

2.1 Vision for IEs

Using machine vision systems for IEs, as opposed to for more general-purpose HCI applications, can change basic assumptions about what types of data the systems are providing. IEs frequently do not require high-precision information, but they require that information be supplied on a time scale comparable to the events they are observing. In addition, people are comfortable interacting in very complex, unstructured environments, so it is reasonable to require that the vision systems be capable of functioning within them as well.



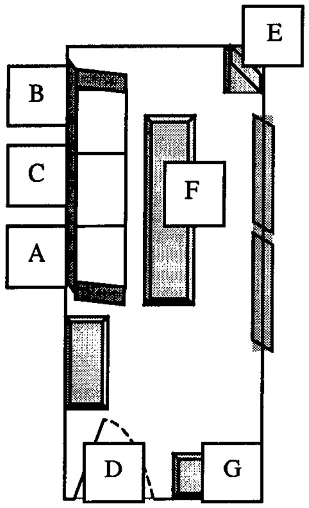


Figure 1A _ A picture of Hal from the doorway with several cameras labeled.

Figure 1B – Hal's floor plan with all camera positions labeled.

Much of the information needed in human-computer interaction is qualitative in nature. For example, it may be necessary to distinguish between a person sitting on a chair, a person standing in front of the chair, and a person standing by the door. This information fits more accurately into a qualitative classification system than into a system that gives only quantitative information (such as position and size).

Since much useful information can be described qualitatively and since qualitative information should be obtainable without a precise geometric model of the world, we insist that precise external calibration be avoided in our vision systems. By minimizing the amount of required external calibration, we expect it should be easier both to modify the configuration of the system and to make it robust with respect to small perturbations, such as a camera being moved accidentally. We view both of these characteristics as essential to making intelligent environments viable in the "real-world" – outside of controlled laboratory conditions.

Information from machine vision systems used in a noisy, unconstrained environment often possesses a large degree of uncertainty. Some sources of the uncertainty include unfavorable lighting conditions and coincidental color matches between objects – for example, a person wearing a green shirt who happens to be sitting in a green chair may seem to disappear to many vision systems. One way to reduce the amount of uncertainty in the system is to replicate the systems and place additional cameras at different locations in the environment. Confusing shadows that are visible for one camera may not be problematic for a second camera viewing the scene from a different perspective, and the overlap between foreground and background objects will likely be different for them as well.

3 Interacting with Hal

Hal is a small room within our lab that doubles as the first author's office (See Figures 1A and 1B). Hal is equipped with microphones, seven video cameras, and a variety of audio-visual output devices that it can directly control. The cameras, microphones, and devices are connected to a cluster of a dozen workstations in an adjacent room. These workstations run a distributed software agent system that controls Hal's operation [2].

Hal was designed to explore a wide range of interactions involving futuristic residential spaces–stressing quality of life–and commercial spaces – stressing information management. We have therefore created applications in Hal that support entertainment, teleconferencing, business meetings, military command post scenarios, and information retrieval. (More in-depth descriptions of these applications can be found in [3,5].) Hal was carefully designed with respect to its intended applications [3]. Because we were particularly interested in creating teleconferencing and meeting recording systems within Hal, cameras had to be strategically placed to obtain useful views of Hal's inhabitants. Many visually based room tracking systems, such as [1], use fixed, ceiling-mounted cameras in very large rooms, because they can obtain

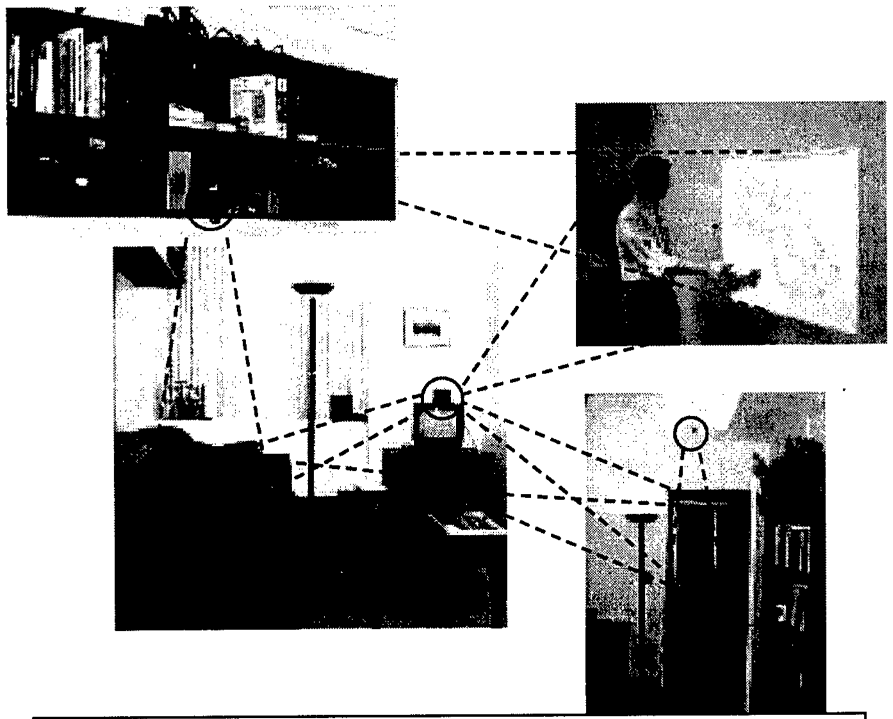


Figure 2 – Hal's layout. Cameras are circled in the image, and sections of each camera's field of view are delineated with dotted lines.

bird's eye views and are less sensitive to visual occlusion. However, cameras placed in or near a ceiling cannot typically obtain images of people that are suitable for other people to watch. It was therefore necessary to place cameras inside Hal that were able to obtain high-quality images of its inhabitants' faces.

We also wanted these cameras to track people during teleconferencing, so that users could freely move about the room; people who are teleconferencing should not need to concern themselves with staying in the field of view of a particular camera. However, the proximity of the users and teleconferencing cameras made it likely that people might quickly move out of an individual camera's field of view. We therefore opted for hybrid approach, combining fixed bird's eye cameras outfitted with wide-angle lenses – that could see large sections of Hal – with steerable camera that could quickly focus on much smaller sections of the room. Our goal has been to allow these systems to intercommunicate to overcome their inherent individual limitations.

Of Hal's seven cameras, two are dedicated to a special purpose vision system that allows a user to manipulate a computer's mouse cursor by pointing a laser pointer at either of two projected video displays. These are labeled A and B in the figure, and will not be discussed further in this paper. Cameras C & D are fixed video cameras with wide-angle lenses. Camera C is mounted in an overhead bookshelf and views the couch and coffee table areas. Camera D is ceiling mounted and overlooks the doorway. Cameras E, F, and G are Sony EVI-D30 steerable cameras under computer control. Each of these cameras has a pan range of approximately 180 degrees and tilt range of approximately 90 degrees. Figure 2 contains several views of Hal and illustrates how the fields of view of the cameras overlap.

The following scenario is an amalgam of several illustrative interactions that currently run inside Hal. (Capital letters in the following refer to cameras, e.g., "E" is Camera E in the floor plan.) The first author walks inside Hal. Hal sees him via D and rotates whichever of E or F is not otherwise occupied to obtain an image of his face. Hal can currently detect the presence of faces in images but cannot yet distinguish people by them. However, seeing a face in the doorway increases the confidence that someone actually just entered the room. Also, this image can be videotaped or transmitted during a teleconferencing session so that remote participants can see who has entered. Hal says, "Good evening Michael, you have three messages waiting." Hal displays the messages on one of the projected displays. As the user walks to the couch and sits down, Hal will attempt to track him via E or F. Once the user is seated on the sofa, Hal sees him via C and keeps F focused on his upper torso. As the user moves around the sofa/coffee table area, Hal tracks him with F. C is used to help steer F in case the user moves out of F's field of view. If the user moves too close to F and Hal can no longer obtain a good image of him, Hal will switch to E (or eventually perhaps G, which due to temporary hardware limitations, does not currently interoperate with E). Next, the user lies down on the sofa. Supposing his clothing is quite dark, he blends into the background so Hal isn't sure if he has actually lay down. Hal will then search for his face via C and will sweep F to make sure it cannot see him sitting up. Hal will then infer that he did indeed lie down; it will next ask if he is going to relax for a little while. If the user answers affirmatively, Hal will dim the lights, close the drapes, and put on Mozart softly in the background. After Hal dims the lights, it will update its model of what the room looks like.

4 Implementation

The current implementation of Hal's vision systems contains three major parts. At the lowest level is a common image processing library that processes the video streams from each of Hal's cameras (with the exception of A and B, as mentioned above). In programmer's parlance, the same "code" is running over all the video streams. The data obtained from each stream, in the form of high dimensional feature vectors, is then passed up to a higher-level system of Image Processing Agents that build statistical cluster-based models to uniquely classify the events seen from each camera's viewpoint. Recursively, these single-camera events are then themselves temporally clustered, creating a global qualitative model of how a single perceptual event within Hal is simultaneously perceived by any of the cameras that can see it. For example, when Hal builds a model of someone walking through the doorway, this model contains what it expects Cameras D, E, and F would see if they were watching someone do so.

These multi-viewpoint models can then used to increase confidence that events are being detected correctly. For example, consider the case where Hal thinks someone is entering the room because of data obtained by Camera D. If Hal knows that this event corresponds to a certain *(pan, tilt)* value for Camera E, Hal can "borrow" this camera for a moment (presuming it is being used for something else) and turn it to that orientation to supplement that data obtained from Camera D.

We now discuss each of the vision system's levels in turn.

4.1 Low-level image processing

Each camera is connected to a separate workstation equipped with a frame grabber and running a low-level image-processing library. This library transforms each incoming image into a feature vector containing background difference, skin detection, and face detection information. (Optical flow values can also be calculated for each video stream, but this is currently only done for Camera D.) These vectors are then available for the Image Processing Agents described in the next section. The low-level libraries process images somewhere between 10-30Hz, depending on what they are viewing. For example, an image containing a face takes longer to process than an image without one.

We discuss here the three primary components of the feature vector and the optical flow calculation.

Background Differencing

Although background differencing itself is a relatively simple technique, the fact that the system uses steerable cameras adds significantly to the complexity of the process. Although it is straightforward to create an array of background images for a small number of predetermined *(pan, tilt)* values for a steerable camera, it is not feasible to capture and store a background image for each of the thousands of possible pan-tilt orientations. It is possible, however, to completely span the camera's viewable world using only a few dozen images. By then performing appropriate coordinate transforms, it becomes feasible to dynamically synthesize a background for an arbitrary camera orientation. We currently use eighteen images (two rows by nine

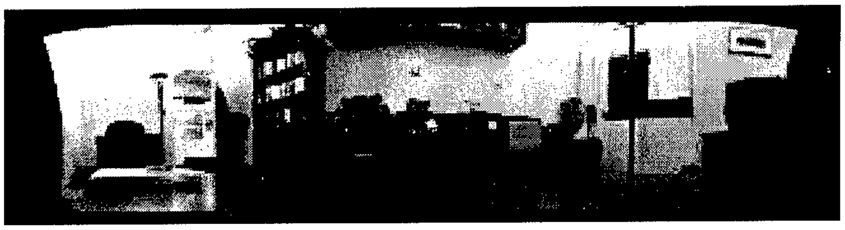


Figure 3 – Sample panoramic background generated by Camera F from 18 source images and stored in transformed pan-tilt space. Pan range = 180° and tilt range ≐ 60°.

columns of slightly overlapping images) to span each camera's range of 180 degrees of pan and 60 degrees of tilt. The system uses these images to create one large "panoramic" background (Figure 3), which is stored in the transformed pan-tilt space instead of x-y coordinates of a certain image position.

A myriad of issues must be dealt with while building panoramic backgrounds. This includes selecting an appropriate scaling for the background that avoids undersampling the images. The coordinate transform from camera image x-y space to panoramic background pan-tilt space also depends on several parameters associated with the camera. Space limitations preclude detailed discussion, but these include the camera's aspect ratio, focal length, radial distortion, and precise axis of rotation. For example, the cameras used by the system are not ideal; they exhibit significant but consistent radial distortion due to imperfections in the lenses. This type of distortion can be modeled by determining a "center of distortion" and by radially expanding or contracting the image according to an empirically determined power series. Although it was decided that empiric calibration would be most expedient, at some point it may be easier to automate much of the calibration using the procedure described in [12].

Skin Color Detection

The second algorithm used by the low-level system is a skin-color detection algorithm that operates independently on each input pixel. Because skin color is determined by a combination of the red hue in blood and the yellow-brown hue of melanin, the hue of skin is restricted to a narrow range of hues. To detect input pixels in this narrow range of hues, the input RGB values were transformed into a log-opponent color parameterization according to a variant of the method described in [6].

Face Detection

The third algorithm that is applied to the input is a face detection algorithm. The algorithm is a ratio template algorithm developed by [11] and used in the Cog humanoid robot [10]. This algorithm is quite computationally efficient when compared to neural-network approaches, and it performs reasonably well in a cluttered environment. These features make it suitable for human-computer interaction scenarios.

Optical Flow

A correlation-based optical flow algorithm compares localized patches of an image to patches of a subsequent image with a range of possible offsets from the original patch. The best match in the subsequent image is assumed to represent the location of the corresponding patch in the new image, and the spatial offset between patches in the two images corresponds to the optical flow. Optical flow information is assumed to be the same as the motion field of the image. This information is therefore useful for detecting transient events such as entries and exits. These events happen too quickly to be reliably detected by the other low-level image processing components.

4.2 Local Event Processing – Image Processing Agents

The next level of the system is comprised of *Image Processing Agents*, which uniquely connect with the different low-level libraries processing the incoming video streams. Each Image Processing Agent receives a stream of feature vectors corresponding to the visual input of the camera with which it is connected. In order to reduce the communication bandwidth between the low-level systems and the Image Processing Agents, which generally all run on different workstations, background difference and skin color information are not transmitted for each pixel. Instead, the image is divided into a 10x10 cell grid, and average values are reported for each cell. Currently, up to two faces can be reported for each image, where a face notification consists of its size and location in pan-tilt space. (The limit of two faces was arbitrarily chosen to simply the communication protocol and can easily be increased.)

Each Image Processing Agent therefore receives a 209 dimensional feature vector, consisting of 100 background difference values, 100 skin-tone match values, 6 faces values, the center of the image in pan-tilt coordinates, and a measure of the overall brightness of the image, at a frequency of somewhere between 10-30Hz.

The agent uses these vectors to build statistical models of the events its camera is capable of detecting during a training session described in the next section. After building these models, it attempts to classify future incoming data to determine if any of the previously learned events reoccur. When they do, this information is passed to the global event processing system described in the next section. The agent is also responsible for several other tasks. The first of these is maintaining the quality of its background image, against which differences are computed to perform foreground object segmentation. Persistently unrecognized phenomena, such as a region being occluded without skin-tones being present, can lead the agent to incorporate that region into the background model. Agents that are connected to steerable cameras can also steer the cameras to track moving objects that catch their attention, even if these objects move away from the location of a previously learned event. The dynamic background synthesis performed by the low-level library makes this behavior possible.

The Image Processing Agent constructs a window W of approximately two seconds worth of the most recently received feature vectors. We will call the size of this window k, so $W = [\bar{v}^{i},_{...}\bar{v}^{i+k}]$. It uses this window to construct and subsequently update two special vectors after each new feature vector arrives. The first is an mean vector $\bar{\bar{v}}_j = mean(\bar{v}_j^i,...,\bar{v}_j^{i+k})$. The other is defined over the standard deviations of the dimensions in the feature vectors, $\bar{\sigma}_j = \sigma(\bar{v}_j^i,...,\bar{v}_j^{i+k})$. Event learning and classification happens over these vectors, rather than over the incoming feature vectors directly.

We define a *local event* E as a tuple $(\bar{\bar{v}}^i, \bar{\sigma}^i)$ of vectors constructed from a window *observing* the event. By this, we mean that the physical event corresponding to E was ocurring in the camera's foveal area while the window was being constructed. Each agent builds up a set of local events, LE, that it is capable of distinguishing during a training process discussed in the next session. (Each agent also constructs at least one "null" event.) Once the agent has been trained, it constantly attempts to classify the event "contained" in the window as one of its previously learned events.

To determine the similarity between a currently observed window and a previously observed event, we define the *distance* between a window W (with $\bar{\bar{v}}$) and a local event $E_j = (\bar{\bar{v}}^j, \bar{\sigma}^j) \in LE$, letting $n = |\bar{v}|$, as:

$$distance\,(W, E_j) = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} \frac{\left|\bar{\bar{v}}_i - \bar{\bar{v}}_i^j\right|}{\bar{\sigma}_i^j + \varepsilon}$$

This function very much resembles a renormalization according to a t-distribution with zero mean and unit variance. However, that is not our intended use for it. Rather, we are using it as a means to calculate a weighted distance between what has currently been observed in the window W and some previously observed event E_j, where dimensions are inversely weighted in proportion to how noisy their signals are. The ε factor is used to avoid difficulties in case a particular dimension has zero variance.

The agent selects $\min_{E_j \in LE}(distance(W, E_j))$ as the most likely event to be occuring. However, events can be partially activated, in that multiple explanations for whatever is currently being observed will be passed on to the global event system discussed in the next session. It is expected in this case that data from other cameras will help disambiguate the room-level event actually being observed. If an agent is asked to verify whether a local-event is occuring as part of this determination, it will attempt to sense the event by forcing whichever feature-vector dimensions it has control over to take on the values associated with the event in question. At present, this is limited to pan-tilt values corresponding to the local event; asserting these values causes the low-level library to move the camera to that orientation.

4.3 Global Event Processing

Global events correspond to qualitative models of how a single perceptual event within Hal is simultaneously perceived by all of the cameras that can see it. There are two mechanisms through which global events are created and/or updated. The first is through an explicit training scenario in which Hal guides users through a series of interactions. The second is a more complicated mechanism where Hal searches for statistically significant temporal co-occurrences of events and infers they are related.

Hal is built on top of a subset of a Room-Perception Application Programming Interface (API) that describes the types of events an abstract Intelligent Environment might sense going on within it, which we simply call *room events*. People writing applications for Hal interface with the perceptual systems via this API, so they can, for example, write code that refers to someone sitting on the middle of couch without having any knowledge of how the perceptual systems work or how this event is recognized by the room. The subset of the API Hal implements has a hierarchy of events within it, such as "Room Enter," "Room Exit," "Sitting Down, "Sitting on Couch," "Standing by Object," etc.

When Hal is initially configured, it leads the user through a training scenario in order to learn what it is like to observe these events. For each fixed camera, this is simply a matter of determining whether there is an unrecognized event in its field of view. The steerable cameras, however, are swept through their range of pan-tilt values in order to locate a new event. Once they have found and centered it, they construct a local event model as described above to represent it. For example, training the "Room Enter" event occurs as follows. Hal says, *"Hello, please enter the room."* The user then walks inside and stands in the doorway. Hal finds them via Camera D and says, *"Please remain still for one moment."* Hal then sweeps the steerable cameras looking for an unrecognized event. As the cameras find and center the user, Hal says *"Aha! I've found you! Thank you for training."* A more sophisticated approach might involve the vision systems exchanging color histogram information to insure they are indeed looking at the same event. However, we assume that simultaneous new room events do not occur during training, so there is currently no need to resolve them.

By determining which local events are constructed and/or activated while training a room-level event, Hal builds qualitative models of the room event with reference to them. If the user does something which triggers the associated local-events of a room event, Hal triggers the appropriate Room-Perception API event, such as "Standing by bookcase," and sends it to any applications that have registered they are interested in such events.

A natural extension of the system at this level would be to build a Markov model of transitions among events from observations of typical usage patterns. This model could be used during event transitions to turn steerable cameras preferentially towards events that are more likely to occur, before they actually happen.

Room events can also be partially activated, meaning some of their composite events have been triggered but not with sufficient confidence to actually be sure the event has taken place. Hal will then try to assert the non-active local events according to the mechanism described above to see if they are currently taking place. Simple statistical confidence intervals over these local events are then used to determine whether something of interest has actually happened. Some care must be taken with this mechanism to insure that events that are transient, such as someone walking through a doorway, have not actually completed by the time the steerable cameras turn to check for them. For this reason, the overhead camera in the doorway produces only optical flow information at 30Hz, so that its local-events are activated as quickly as possible. Within less than .5 seconds of someone walking through the doorway, whichever steerable camera is free to verify the event is already turning to look at it.

We are also interested in using this mechanism to connect the room's visual perception with its other sensory modalities. For example, we have empirically determined that people tend to speak about objects they are closer to and therefore we dynamically bias Hal's speech recognition models with information obtained from its tracking system. These biases are currently hand-coded, but it seems feasible to learn them via temporal correspondences in a manner very similar to how Hal's visual event acquisition currently operates.

References

1. Bobick, A.; Intille, S.; Davis, J.; Baird, F.; Pinhanez, C.; Campbell, L.; Ivanov, Y.; Schütte, A.; and Wilson, A. Design Decisions for Interactive Environments: Evaluating the KidsRoom. *Proceedings of* the *1998 AAAI Spring Symposium on Intelligent Environments*. AAAI TR SS-98-02. 1998.
2. Coen, M. Building Brains for Rooms: Designing Distributed Software Agents. In Proceedings of the Ninth Conference on Innovative Applications of Artificial Intelligence. (IAAI97). Providence, R.I. 1997.
3. Coen, M. Design Principles for Intelligent Environments. In Proceedings of The Fifteenth National Conference on Artificial Intelligence. (AAAI98). Madison, Wisconsin. 1998.
4. Coen, M. (ed.) Proceedings of the 1998 AAAI Spring Symposium on Intelligent Environments. AAAI TR SS-98-02. 1998.
5. Coen, M. The Future Of Human-Computer Interaction or How I learned to stop worrying and love My Intelligent Room. IEEE Intelligent Systems. March/April. 1999.
6. Forsyth, D. and Fleck, M. Finding Naked People, European Conference on Computer Vision, Volume II, pp. 592-602. 1996.
7. Lien, J., Zlochower, A., Cohn, J., Li, C., and Kanade, T. Automatically Recognizing Facial Expressions in the Spatio-Temporal Domain. *Proceedings of the Workshop on Perceptual User Interfaces (PUI'97).* Alberta, Canada. pp.94-97. 1997.
8. Lucente, M.; Zwart, G.; George, A. Visualization Space: A Testbed for Deviceless Multimodal User Interface. *Proceedings of the AAAI 1998 Spring Symposium on Intelligent Environments*. AAAI TR SS-98-02. 1998.
9. Mozer, M. The Neural Network House: An Environment that Adapts to its Inhabitants. *Proceedings of the AAAI 1998 Spring Symposium on Intelligent Environments*. AAAI TR SS-98-02. 1998.
10. Scassellati, B. Eye Finding via Face Detection for a Foveated, Active Vision System, Proceedings of the National Conference on Artificial Intelligence, Madison, WI. 1998.
11. Sinha, P. Object Recognition via image invariants: A case study, Investigative Opthalmology and Visual Science, 35, pp. 1735-1740. 1994.
12. Stein, G. Internal Camera Calibration using Rotation and Geometric Shapes, Master's Thesis, Massachusetts Institute of Technology, Cambridge, MA. 1993.
13. Stiefelhagen, R., Yang, J., and Waibel, A. Tracking Eyes and Monitoring Eye Gaze. *Proceedings of the Workshop on Perceptual User Interfaces (PUI'97).* Alberta, Canada. pp.98-100. 1997.
14. Want, R.; Schilit, B.; Adams, N.; Gold, R.; Petersen, K.; Goldberg, D.; Ellis, J.; and Weiser, M. The ParcTab Ubiquitous Computing Experiment. Xerox Parc technical report.
15. Weiser, M. The Computer for the 21st Century. *Scientific American*. pp.94-100, September, 1991.

Designing for Local Interaction

Johan Redström, Per Dahlberg*, Peter Ljungstrand and Lars Erik Holmquist

PLAY: Applied research on art and technology and *Mobile Informatics Research Groups
The Viktoria Institute, Box 620, SE-405 30 Göteborg, Sweden
{johan,dahlberg,peter,leh}@viktoria.informatics.gu.se
http://www.viktoria.informatics.gu.se/

Abstract. Much development of information technology has been about reducing the importance of distances and user location. Still, many important activities and events are of local nature, for instance serendipitous face-to-face communication. In order to support such communication, as well as other examples of local interaction, we have developed three prototypes all based on wireless short-range communication. The prototypes are functionally self-contained mobile devices that do not rely on any further infrastructure, making the system inexpensive, flexible and easy for users to manipulate. In these experiments, the limited communication range is not conceived as a problem, but rather as a property that can be explored. We present and discuss the Hummingbirds, Generalised Hummingbirds and the NewsPilot, as well as the implications of this approach for human-computer interaction design

1 Introduction

Information is generally not propagated very far from its origin. Signs can not be read if they are not within sight, signals can not be heard unless within hearing distance, and so forth. In this way, the limitations of our perceptual systems in combination with certain properties of information propagation in physical space (i.e. different kinds of carrier waves travelling through obstacles like walls, floors and outdoor topology) can be said to act as information filters. In order to take part of different sources of information, we have to move around in the environment and in order to talk to each other we have to be co-located.

These limitations are shortcomings of physical spaces that we have been trying to eliminate by the use of information technology. The telegraph made it possible to send messages over long distances, the telephone enabled persons at different locations to speak with each other, and more recently the computing industry introduced us to global networks that make it possible to instantly communicate and share information with people all over the world. The introduction of mobile devices that are carried by their users at almost all times, e.g., pagers and cellular phones, have decreased the importance of location even further.

Still, at times proximity is a rather good measure of relevance. We tend to place important objects near us or near the place we are going to use them. Documents and books lying on someone's desktop are more likely to be related to current work than documents placed in filing cabinets or bookshelves. Considering almost ubiquitous resources (at least in office environments) such as power outlets, water taps or rest rooms, proximity is often the main criteria for relevance. We also move around in our environment in order to get a chance to talk people etc. [3, 4, 30].

The usefulness of location as a constraint for information distribution is perhaps best seen when it is removed: now when we can contact almost anybody any time and instantly access information everywhere we are beginning to experience information overload [21] and communication overflow [19]. Given the apparent importance of local interaction, such as face-to-face communication, and local mobility or "roaming" in search for people and resources, rather little has been done to support it [cf. 3, 4, 30].

In this paper, we describe our explorations of the usefulness of proximity as a constraint for information distribution, starting with development of support for awareness of co-located people. We will begin with presenting related work and then report from three projects all aimed towards supporting local interaction. We conclude with a discussion of our experiences and outline their implications as well as future work.

2 Background

2.1 Local Interaction

Although informal communication *per se* is not our main interest here, research on this topic presents a number of relevant themes. In occurring definitions of informal communication, it is clear that local interaction plays an important part. For instance Whittaker *et al.* [30] uses a wide definition as "taking place synchronously in face-to-face settings"; Fish *et al.* [11] mean that while meetings are pre-planned with a predefined agenda, informal communication is a social event, work related or not, that takes place ad hoc when there is an opportunity for communication. The importance of being co-located has been reported in several cases. Relevant studies include Bergqvist *et al.* [4] who studied ad hoc mobile meetings in a work place; Covi *et al.* [7] who studied the effects of dedicated project rooms; Fitzpatrick *et al.* [14] who discussed the difficulties in designing co-operative buildings with support for awareness and serendipitous interaction for distributed groups and Whittaker *et al.* who argued that physical proximity is crucial for informal communication [30].

Extensive research has been done on how to support informal communication using IT. However, in most cases the aim has been to support distributed rather than co-located groups [cf. 9, 12, 14, 20]. The rationale for this is that many incitements for occasional communication are lost when people are not co-located. Thus, support for awareness about peoples whereabouts that could compensate for not having corridors, lunchrooms etc. as sources of such information have been developed. The notion of proximity has also been used as a metaphor in virtual environments designed for social interaction, for instance in Chat Circles [26] and FreeWalk [20]. One of the main differences between these projects and the work presented here is that in a face-to-face setting, the technology does not *mediate* the communication. Therefore, our focus has been on how to *support* communication, for instance by means of providing relevant information.

Another aspect of informal communication is that it is often serendipitous. Hence, technology should support on-the-fly communication, without any need for time-consuming and complex actions on behalf of the user. There are several related attempts to support awareness of colleagues in an office environment [cf. 25, 28]. These systems differ in that they rely on a fixed infrastructure at a number of specific locations, meaning that they only support spontaneous meetings at certain places. Our aim was to support such communication regardless of any specific locations.

2.2 Ad Hoc Networks and Context-Aware Computing

Ad hoc networks are self-organising wireless networks composed of mobile nodes that do not require a stationary infrastructure. They are designed to be rapidly deployed to provide robust communication in a variety of environments, which often lacks a supporting infrastructure [15]. Unlike the work presented in this paper, the objective is to allow for communication between all devices, regardless of the present location. Current research on ad hoc networks is highly technical, mostly about network protocols, rather than taking social considerations or novel applications into account.

Besides communicating, the devices have to make use of what information they send and receive in order to support local interaction. The term "context-aware computing" was introduced as a part of the ParcTab ubiquitous computing experiment [28] to describe mobile and wearable systems that collect data from their environment and use it to adapt their behaviour [1, 24, 28]. A system is said to be context-aware if it keeps track of any aspect of its present context, most commonly location [2, 27, 28]. Thus, being "context-aware" does not imply that the system in question is aware of all aspects of its context.

The prototypes described in this paper are not aware of their absolute position, but only how they are positioned in relation to other devices. A similar approach has been used in a number of cases, perhaps most notably in the LoveGety [18], a recent commercial success in Japan. The LoveGety is a small, wirelessly communicating device that detects other LoveGetys within a certain range, in order to support social encounters between people. Other examples of related systems include the Thinking Tags [6] and GroupWear [5], which tell about relationships between people engaging in face-to-face conversations.

3 Experiments

Beginning to explore the possibilities in designing for local interaction, the field of informal face-to-face communication seemed to be an interesting domain due to its dependence on local interaction and serendipitous communication. We have developed a number of applications all using short-range radio-transceivers (Fig. 1). Below, we will describe the Hummingbirds, the Generalised Hummingbirds and the NewsPilot.

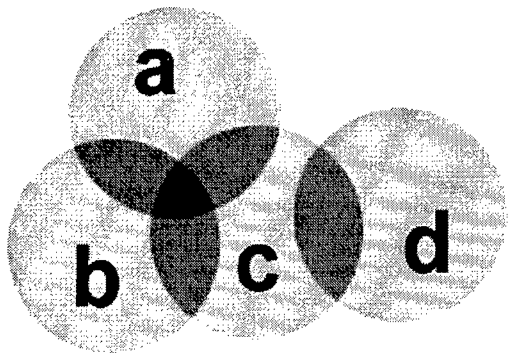


Fig. 1. Figure illustrating four devices, their relative position to each other and what information is available to each of them. The circles represent their respective communication range. In this case, A and B both have access to information distributed by A, B and C; C has access to A, B and D; D has access to C.

The range of the communication varied from about 100 meters in the case of the Hummingbirds, to approximately 10 meters in the NewsPilot, and was deliberately chosen to suit each application.

3.1 Hummingbirds

The Hummingbird [17] is a small wearable device equipped with a short-range radio transceiver, through which it broadcasts its identity and receive information about other Hummingbirds in the vicinity. The devices are functionally self-contained, i.e. non-dependent of surrounding infrastructure. The overall objective is to support awareness of "who's around" within an established group of people. Whenever two or more Hummingbirds are close enough to communicate, the devices give a subtle audio signal and display the identity of the other devices in the proximity. In this way, it is possible for users to know which other Hummingbird users are in the proximity.

Inspired by what is often within "shouting distance", the communication range is set to approximately 100 meters (depending on the number and nature of obstacles like people, walls etc.). The rationale for this range is that as the Hummingbirds do not show in what direction other users are located, the space in which to search for them must not be too large if information about their presence should be useful. The reason for not presenting directional cues on the Hummingbird is partly due to technical difficulties, but more importantly the wish to make the devices as unobtrusive as possible, regarding their use as well as the perception of them. It is important that the Hummingbirds are not perceived as surveillance devices.

In studies of user experiences we have found that the Hummingbird is particularly useful in situations where a group of users are outside their normal environment, e.g., when travelling [17, 29]. The Hummingbird experiment has shown that for mobile

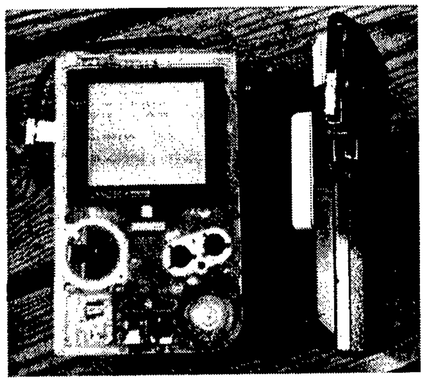


Fig. 2. Picture showing the Generalised Hummingbirds, i.e., GameBoys fitted with radio transceivers.

users, it can be valuable just to have the knowledge that other users are in the vicinity, although it is not possible to use the Hummingbirds to mediate communication.

3.2 Generalised Hummingbirds

The results from the experiments with the Hummingbirds inspired a more general platform. The hardware is based on the Nintendo GameBoy, a hand-held video game. The GameBoys are fitted with small radio transceivers that are connected to the devices' serial ports (Fig. 2). The range of communication is about 50 meters. The reason for decreasing the range compared to the original Hummingbirds, is the fact that more sources of information would be used and that information therefore had to be filtered to a greater extent. Modified game cartridges are used for installing custom software.

The Generalised Hummingbird enables users to give their devices arbitrary names, making identification easy. As with the original Hummingbird, Generalised Hummingbirds are only able to communicate by means of sending and receiving their digital signatures (i.e. their "names"). In order to support events over a wider time frame than the present, the names received are displayed as being in one of two states: "active" when the Generalised Hummingbird is currently picking up the signature in question, and "inactive" when the device recently has picked up the signature but ceased to do so (within the last half an hour or so). This enables users to see a trace of what has happened recently.

Applications. The Generalised Hummingbird enables users to obtain further awareness about activities in their near surroundings using both the "trace" functionality and the possibility to associate devices not only to people, but to places and artefacts as well. For instance, when a user enters a building, an ordinary Hummingbird will pick up what other devices are present, but not which have been there recently. However, if a user places a stationary Generalised Hummingbird at a certain location in the build-

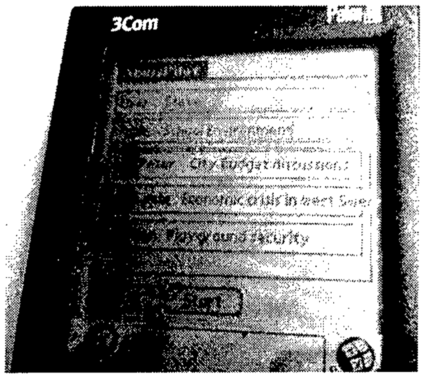


Fig. 3. Picture showing the NewsPilot (main screen), i.e., a 3COM Palm III fitted with a radio transceiver similar to the ones used with the Generlised Hummingbirds (Fig. 2).

ing, its display will show what signatures it has received recently, thus showing a trace of recent activities at that location.

Certain places, like the corridors, act as informal meeting places [cf. 3, 30]. In order to make information about activities in the lunch room available, users can connect a Generalised Hummingbird to a movement detector, ensuring that whenever there is any activity in that room the device is turned on and, thereby, broadcasting its signature. Correspondingly, the activity of some artefacts might be of interest. For instance, it is possible for users to monitor the availability of fresh coffee using a Generalised Hummingbird connected to a coffee machine, so that whenever fresh coffee is available, a device named "Coffee" becomes active.

This experiment illustrates that it is possible to add a variety of information sources to the network using both mobile and stationary devices, without making the human-computer interaction any more complex than in the original Hummingbird example. Adding new information sources is not any more difficult than moving them into the place in question, and if the devices are to be used for keeping track of some activity, ordinary, affordable and easy-to-manage solutions can be used.

3.3 The NewsPilot

The next step of development is the NewsPilot [8], a design based on implications from an empirical study at a Swedish radio station working with broadcast news, conducted by the MobiNews project at the Viktoria Institute. The NewsPilot is based on the 3Com Palm III PDA (Personal Digital Assistant) fitted with a radio transceiver (Fig. 3). The communication range is decreased even further compared to previous experiments, to about 10 meters, since that was a more appropriate range for how far away proximity was relevant at the radio station.

People. The journalists at the station relied on a large number of information resources to select and compile news stories, e.g. local newspapers, television, fellow journalists, etc. The discussions among the colleagues were an important part of the local news dissemination. Often, several journalists had been involved in various related topics, making for fruitful discussions. To help initiating such discussions each NewsPilot user is able to enter a short message stating what he or she is currently working on. By sending out the message together with name of the user, users can obtain information not only about which colleagues are in the proximity, but also what they are working on.

Places. The second important finding during the study was that it seemed like different types of information were important at different locations. For instance, there was a table and a shelf with newspapers in the centre of the office used for reading and annotating recent newspapers. When a journalist attended this location, he or she was generally interested in getting information about related stories produced both internally and externally. By fitting transceivers to the walls, messages can be distributed to NewsPilots at specific locations. At the newspaper-table, the task-message from the NewsPilot is received by a wall-mounted transceiver connected to a stationary PC, which is used to search local network resources for relevant information. An additional server is used as an interface between the transceivers and network resources. Short messages about findings are sent back to the NewsPilot and presented to the user. Each message contains an abstract and information on where the full story can be retrieved. Although a user hardly wants to walk to a specific location just to filter out information, if viewed as a complement to traditional searching and browsing, location-based filtering might assist the user in her work.

4 Discussion

4.1 Supporting Serendipitous Communication in Face-to-Face Settings

There seems to be at least two reasons for initiating occasional communication: either (1) that at least one of the participants has a question or subject that she or he wants to discuss, or (2) that the situation as such is an incitement for a conversation. In the first case the subject of the conversation is "known" before the conversation takes place; in the second the subject will be chosen according to the situation more or less spontaneously. While these two cases are superficially similar, the underlying properties differ and will have to be acknowledged in the design of a supporting system.

The first situation is in many respects similar to more "explicit" communication such as phone calls or e-mail, as there are a rather well defined subject and a target person. The main difference is that the property of talking face-to-face is so valuable, that other variables, e.g., when or where to talk, can be left open. A person might choose slightly different strategies in order to catch a talk with her target. Staying in her room will probably mean fewer encounters with other people including the target, than walking around in the office more or less searching for the target. There seems to be a continuous scale of more or less explicit actions in order to make the meeting happen. The key factor for initiating such a conversation is presumably knowledge that the target person is in the vicinity and available for a chat. This is probably one reason why awareness about other people's whereabouts seems important. However, if the proposed discussion target is not available for the time being, there is always a risk that the idea sinks into oblivion. We all need a reminder from time to time, and this is one reason why calendars and to-do lists (electronic or paper-based) are so popular. A future implementation of a support for this first kind of situation might therefore be a context sensitive "to-do-list" that reminds its users when the appropriate context for completing the task turns up [cf. 23].

While the first situation bears on one of the participants having an interest in talking about something, the second one seems to arise out of the interest in talking as such. This might be due to being in a place where social interactions commonly take place, or because the situation as such is suited, or demands for that matter, that people initiate conversations. Consider for instance the following common scenario: a person A walks down the corridor and meets another person B. As they begin to talk A notices that B carries a certain book that she is reading too. As a result they begin to discuss the book, and both A and B might get useful information about aspects not thought of, related references etc. Unless B had carried that book, the discussion might never have taken place. This useful sharing of experiences among co-workers obviously does not rely on one of the participants already knowing what to talk about, but on the situation as such in combination with certain resources present. Thus, a support for this kind of situations will not be in the form of a reminder service, but rather some way of presenting relevant information based on criteria such as the participants present tasks, interests, projects etc. Selection of such information could for instance be a matching between current interests of the users in order to find out the least common denominator and present relevant information, functioning in a way similar to carrying around books in the example above. The NewsPilot was an attempt to provide such a tool for journalists at a news agency.

As we have shown, systems based on locally communicating devices can be used in both cases. The Hummingbird supports an awareness of who is in the proximity assisting a person that has a predetermined wish to communicate. However, it does not provide any help for picking a topic once the parties have met. The Generalised Hummingbird and the NewsPilot provides similar awareness, but with different ranges. Further, the NewsPilot can support persons in choosing a topic to talk about by providing information on what topics other participants are working on. Most of this can be, and have to some extent been, realised using other techniques than local communication between functionally self-contained devices. However, there are some advantages with the strategy employed here that are interesting from a human-computer interaction point of view, some of which will be discussed below.

4.2 Implications for Human-Computer Interaction Design

Let us first sum up a few of the properties of local interaction between devices that we have tried to exploit in order to design easy-to-use technology. The radio transceivers enabled us to use the limited range of communication to create something similar to an adaptive location-based information filter. The communication between the devices was established on the fly, and did not require any explicit actions on behalf of the users. Since the devices are functionally self-contained, users did not have install, configure and maintain any additional infrastructure, except for the stationary information servers used in the NewsPilot, making the systems as a whole easy to manage, move and manipulate.

Our experiences suggest that the principle of *proximity as a constraint for information distribution* can have a wider applicability than what has been presented here. In the Generalised Hummingbirds, we introduced sources such as places and artefacts, and in the NewsPilot users could have information sent to their PDAs when visiting a certain place. The usefulness of these additions suggests that it is interesting to support local interaction not only among people, but with other "resources" in the vicinity as well. For instance, we can use local communication between devices in order to let the user combine their respective functionality in the manner Norman suggested "information appliances" to behave [22]. We can also imagine a scenario where users can create computationally augmented, or rather "amplified" [10], environments using "building blocks" that keep their functionality when moved to new locations: if two units work in a certain way together, they will continue to do so when moved somewhere else.

This stands in contrast to most implementations of ubiquitous computing, in which the rather simple devices users interact with, rely on an advanced "hidden" infrastructure. As the behaviour and functionality of their devices will change radically depending on what hidden resources or infrastructures are there to back them up, users will have to care about something that originally was designed to be invisible. This is not very fortunate, considering the aim to hide complexity away from the user. The absence of such hidden resources might help users to build, manipulate and understand their computational environments. Of course, such strictly local networks of devices will not be able to solve all the problems dealt with in ubiquitous computing and intelligent environments, but they might serve as an interesting complement.

4.3 Spaces and Places

When discussing wirelessly communicating devices, the notion of range is often used to describe the size of a cell, i.e. a certain area with a set of devices (people) that all can directly communicate with each other. The term "range" refers to physical distance in space, and it is natural to use this term when discussing spaces. However, the spaces we move about in are, when a social context is applied, perhaps better be understood in terms of *places* [16] or *locales* [13]. A place is the understood reality, e.g. a room, an office or a building, while space only contains spatial measures [16]. Since we are no longer strictly talking about spaces, but rather about perceived places, i.e. a combination of a physical location and a social context, the notion of range becomes somewhat inappropriate.

Consider for instance a group of people sharing a room, where all participants can talk to and hear each other. This can be seen as a communication cell. Suppose we break the group into two smaller groups. Now we have two smaller cells, independent of each other. This makes very much sense when we only use verbal communication,

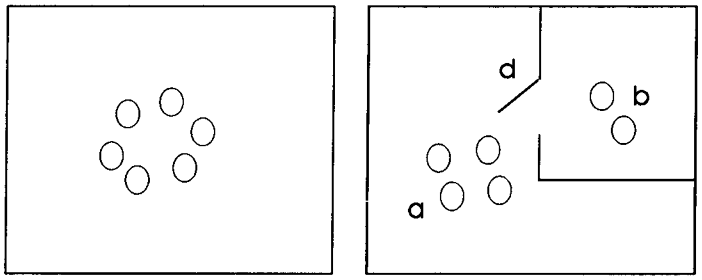


Fig. 4. These figures illustrates the problem of proximity in terms of two different scenarios where different verbal communication ranges apply. In the figure to the left, the relevant cell encompass all six people; in the right one there are two independent groups (a and b). Further, the degree of independency between the two groups of people partly depends on whether the door (d) is shut or open.

but current wireless communication devices would not follow these changes as the size of the relevant communication range changes depending on context. This is even more obvious in the case of dividing spaces into different rooms (Fig. 4). Developing technology that communicates within a given place (or part of a place, if that is more appropriate) instead of communicating within a certain space would add new possibilities to this kind of technological support. In other words, we want a notion of "proximity" that is more complex and incorporates more than just the physical distance between things, for instance in what social context they are located. To complicate things, different applications might want to use proximity differently. Sometimes the actual physical (spatial) proximity is preferred, as in the case with the Hummingbirds, while other situations might require more adaptive techniques. Clearly, much remains to be done in this area.

5 Conclusion and Future Work

We have tried to make the case that limited communication range is an interesting constraint when designing support for local interaction. We have also discussed some implications of the experiences we have had with our prototypes and discussed the ideas that have arisen during this work. As we have tried to argue, there are a number of interesting properties associated with local interaction between devices, for instance aspects such as information filtering and the possibilities of transparent human-computer interaction.

We have mentioned future projects such as dynamic and context-sensitive to-do-lists and supporting people engaging in spontaneous meetings with relevant information that could enhance applications similar to the ones described here. In order to investigate such applications, the design efforts, combined with user studies and evaluations, will continue. Important issues for future projects do not only include exploring new services for local interaction, but also to develop a "smarter" notion of proximity in order to acknowledge properties of space that are of social importance. Some easily perceived cues, like open and closed doors, windows and walls, are within reach. Designs for more subtle properties, such as different groups talking to each other within a certain room, will be harder to achieve. However, having a technology that could cope with such aspects of the context would enhance its usability dramatically, given the purposes proposed in this paper.

The suggested approach will also have to be evaluated on other domains than face-to-face communication. For instance, the easy-to-manage (from certain points of view, that is) nature of systems composed of communicating and functionally self-contained devices can make it easy for users to create and manipulate their own "smart" environments using a variety of devices. Other important issues in future work include security and privacy issues. So far, the need for authentication or a high security level has been rather small due to the nature of the information distributed. However, when developing other applications those issues must be dealt with.

We believe that support for local interaction is an exciting area. Further development in the areas of wireless networks and mobile computing will make many new types of devices that support local interaction possible.

6 References

1. Abowd, D., Dey, A., Orr, R. & Brotherton, J. (1998) Context-Awareness in Wearable and Ubiquitous Computing. Journal of Virtual Reality. 3, pp 200-211. Springer-Verlag.
2. Abowd, G, Atkeson, D., Hong, C., Kooper, J., Long, R. & Pinkerton, M. (1997) Cyberguide. A mobile context-aware tour guide. In: ACM Wireless Networks, 3(5), pp. 421-433. ACM Press.
3. Bellotti, V. and Bly, S. (1996). Walking Away from the Desktop Computer: Distributed Collaboration and Mobility in a Product Design Team. In: Proceedings of CSCW '96, pp. 209-218. ACM Press.
4. Bergqvist, J., Dahlberg, P., Ljungberg, F. & Kristoffersen, S. (1999). Walking Away from the Meeting Room: Exploring Mobile Meetings. To appear in: Proceedings of ECSCW'99, Copenhagen, Denmark.
5. Borovoy, R., Martin, F., Resnick, M. & Silverman, B. (1998) GroupWear: Nametags that Tell about Relationships. In: CHI'98 Conference Summary, pp. 329-330 ACM Press.
6. Borovoy, R., McDonald, M., Martin, F., & Resnick, M. (1996). Things that blink: Computationally augmented name tags. In: IBM Systems Journal, Vol. 35, No. 3&4, pp. 488-495.
7. Covi, L., Olson, J. & Rocco, E. (1998) A Room of Your Own: What Do We Learn about Support of Teamwork from Assessing Teams in Dedicated Project Rooms? In: Proceedings of First International Workshop on Cooperative Buildings (CoBuild'98), pp. 53-65. Springer-Verlag.
8. Dahlberg, P., Redström, J. & Fagrell, H. (1999). People, Places and the NewsPilot. In: Extended Abstracts of CHI '99, pp. 322-323. ACM Press.
9. Dourish, P. and S. Bly (1992). Portholes: Supporting Awareness in a Distributed Work Group. ACM 1992 Conference on Human Factors in Computing Systems, Monterey, CA, ACM Press.
10. Falk, J., Redström, J. & Björk, S. (1999). Amplifying Reality. To appear in: Proceedings of HUC 99 (International Symposium on Handheld and Ubiquitous Computing). Springer-Verlag.

11. Fish, R., Kraut, R. & Chalfonte, B. (1990). The VideoWindow system in informal communications. In: Proceedings of ACM 1990 Conference on Computer-Supported Cooperative Work, Los Angeles, CA, pp. 1-11. ACM Press.

12. Fish, R., Kraut, R., Root, R. & Rice, R. (1993). Video as a technology for informal communication. In: Communications of the ACM 36(1), pp. 48-61. ACM Press.

13. Fitzpatrick, G, Kaplan, S. & Mansfield, T. (1996) Physical Spaces, Virtual Places and Social Worlds: A study of work in the virtual. In: Proceedings of CSCW'96. ACM Press.

14. Fitzpatrick, G, Kaplan, S. & Parsowith, S. (1998) Experience in Building a Cooperative Distributed Organization: Lessons for Cooperative Buildings. In: Proceedings of First International Workshop on Cooperative Buildings (CoBuild'98), pp. 66-79. Springer-Verlag.

15. Haas, Z. & Pearlman, M. (1999) Determining the Optimal Configuration for the Zone Routing Protocol. IEEE Journal on Selected Areas in Communications, Special issue on Wireless Ad Hoc Networks, June 1999.

16. Harrison, S. & Dourish, P. (1996) Re-place-ing space: the roles of place and space in collaborative systems. In: Proceedings of the ACM 1996 conference on on Computer supported cooperative work, pp. 67-76. ACM Press.

17. Holmquist, L. E., Falk, J. & Wigström, J. (1999). Supporting Group Collaboration with Inter-Personal Awareness Devices. To appear in: Journal of Personal Technologies, Special Issue on Handheld CSCW. Springer-Verlag.

18. Iwatani, Y. (1998) Love: Japanese Style. In Wired News, 11 June 1998. Available at: http://www.wired.com/news/culture/story/12899.html.

19. Ljungberg, F. and Sørensen, C. (1998). Are you pulling the plug or pushing up the daisies? In: Proceedings of Thirty-First Hawaii International Conference on System Sciences (HICSS'31), Hawaii. IEEE Computer Society Press.

20. Nakanishi, H., Yoshida, C., Nishimura, T. & Ishida, T. (1996) FreeWalk: Supporting Casual Meetings in a Network. In: Proceedings of CSCW'96, pp. 308-314. ACM Press.

21. Nelson, M. (1994) We have the information you want, but getting it will cost you! Crossroads 1(1), ACM Press.

22. Norman, D. (1998). The Invisible Computer. Cambridge, Massachusetts; MIT Press.

23. Rhodes, B. (1997). The Wearable Remembrance Agent: A system for augmented memory. In: Journal of Personal Technologies; Special Issue on Wearable Computing, pp. 218-224. Springer-Verlag

24. Schilit, B. (1995) A Context-Aware System Architecture for Mobile Distributed Computing. Ph.D. Thesis, Columbia University, May 1995.

25. Tollmar, K., Sandor, O. & Schömer, A. (1996) Supporting Social Awareness@Work Design and Experiences. In: Proceedings of CSCW'96, pp. 298-307. ACM Press.

26. Viegas, F. & Donath, J. (1999). Chat Circles. In: Proceedings of CHI'99, pp 9-16, ACM Press.

27. Want, R, Hopper, A, Falcao, V & Gibbons, J. (1992). The Active Badge Location System, ACM Transactions on Information Systems, Vol. 10, No. 1, January 1992, pp 91-102, ACM Press.

28. Want, R., Schilit, B., Adams, A., Gold, R., Petersen, K., Goldberg, D., Ellis, J. & Weiser, M. (1995). The ParcTab Ubiquitous Computing Experiment. Technical Report CSL-95-1, Xerox Palo Alto Research Center, March 1995.

29. Weilenmann, A. & Holmquist, L. E. (1999) Hummingbirds Go Skiing: Using Wearable Computers to Support Social Interaction. To appear in: Proceedings of Third International Symposium on Wearable Computers (ISWC) '99, San Fransisco, CA.

30. Whittaker, S., Frohlich, D. & Daly-Jones, O. (1994). Informal workplace communication: What is it like and how might we support it? In: Proceedings of CHI '94, pp. 131-137. ACM Press.

The SCD Architecture and its Use in the Design of Story-Driven Interactive Spaces

Claudio S. Pinhanez

MIT Media Laboratory, 20 Ames Street
Cambridge, Massachusetts, MA 02139, USA
pinhanez@media.mit.edu

Abstract. In this paper we examine story-driven interactive spaces and argue that story development requires centralized control of the actions of the characters that inhabit such environments. This argument leads us to propose the *story-character-device (SCD)* architecture that, unlike previous approaches, separates the locus of story control in a distinct software module. To exemplify the use of the architecture we examine an art installation (called *"It"*) that we built according to the SCD model. The development process of *"It"* is then compared to the design of another story-driven interactive space that used similar technology and thematic, the computer theater play *"It / I"*. By looking into the design process of the two environments, we conclude that the SCD architecture allows much more flexibility in the design process of story lines besides bringing an overall simplification of the control system.

1 Introduction

There has been an increasing interest in creating interactive spaces for entertainment with a strong narrative structure or with an underlying story [4, 10, 16, 24]. These systems have employed many different control architectures, each of them with several shortcomings. The goal of this paper is to analyze the fundamental characteristics of story-based interactive spaces and how their characteristics influence the design of their architecture.

From our experience in developing story-driven interactive systems for entertainment (*"The KidsRoom"* [4], *"SingSong"* [22], and *"It / I"* [21]) we have concluded that centralized story control is essential to achieve successful story development. Such a claim is clearly contentious. Perlin and Goldberg [18] as well as Bates et al. [2] built (semi-) autonomous computer-actors used in systems where the story was distributed among characters or seen as the natural result of the interaction between the characters and the user. However, as noted by Langer [12] and Murray [15], well constructed stories require coordination and synchronicity of events and coincidences which we believe can only be achieved by centralized story control.

In this paper we propose a three-level architecture for story-driven interactive systems called *story-character-device* architecture, or *SCD*. The defining characteristic of the SCD architecture is the separation between character and story control. In the SCD architecture the characters are semi-autonomous, receiving commands from the story control module although possessing the ability to sense the on-going action and to coordinate the accomplishment of their goals according to the reality of the story world.

To exemplify the use of the SCD model we examine *"It"*, an interactive story-driven interactive space that recreates from a first-person point of view the story of the computer theater play *"It / I"* [21]. The original play featured an automatic character on stage interacting with a human actor through computer graphics, sound, and light, according to information gathered by cameras. In other words, the stage was transformed into an interactive space. *"It / I"* was originally produced using an almost flat system architecture and therefore, since *"It"* basically portrays the same story, we can compare the design and development process without and with the use of the SCD architecture.

2 Interactive Spaces

In this paper we adopt the term *interactive space* to refer to situations where human beings interact with a computer in a physical space through digital input/output devices such as cameras, microphones, video screens, and speakers. The classic example of an interactive space is the control room of spaceships as portrayed in science-fiction series and movies like *"Star Trek"* and *"2001:A Space Odyssey"*. In these two cases the computer that controls the spaceship is omnipresent, interacting with the ship's crew as an invisible, God-like figure that can either access the most vital data or entertain the occupants with a game of chess.

2.1 Characteristics of Interactive Spaces

Let us examine some common characteristics of an interactive space. First, it is a **physical space**. For instance, we do not consider either virtual reality or videogames as interactive spaces in our definition. In other words, we are interested in systems that are required to assume the existence of atoms, their physical interaction, and the difficulties in sensing real matter (as opposed to checking the state of virtual entities).

We also assume the existence of a **computer system** controlling the interaction, discarding human-controlled environments like a disco club (where the DJ manages the interaction between the patrons and the dance area). Moreover, interactive spaces must be **responsive**, that is, they must acknowledge the attempt of interaction by the users either by answering directly or by producing a detectable transformation in the space.

Another aspect is that agency should be attributed to the space itself and not to particular objects in the space. Perhaps the most characteristic feature of an interactive space is precisely this **omnipresence**. Notice that in the case of entertainment spaces inhabited by virtual or physically represented characters, the agency of the characters has clear locus, but the space must still be the non-localized source of the story, game, or narrative.

Part of the appeal of the idea of interactive spaces is exactly that, perhaps for the first time in history, we can see a space as dialoguing entity. Computers and sensing devices create spaces that are **machines**, that is, perceived as animated and active objects.

2.2 User-Driven Interactive Spaces

We employ the term *user-driven* to characterize interactive spaces where the user directs the direction and goal of the overall interaction. This category, in fact, comprises the majority of the work done in terms of creating actual interactive spaces. The pioneer of the idea is Mark Weiser who proposed the concept of *ubiquitous computing* about a decade ago [25]. In its simplest form, ubiquitous computing advocates the disappearance of the visible entity associated to computational power and its spread through the space and everyday objects.

Alex Pentland has promoted the idea of *Smart Rooms* as a domain of perceptual computing [17]. The most successful result was the *ALIVE* experience [14] where the user could interact with a CG-generated dog by watching herself in a *virtual mirror* — a large video screen showing a mirror-like image of the room with the user and superimposed computer graphics objects. Another example is the *Intelligent Room*, being developed at the MIT AI Laboratory [7], aiming to create a room for crisis management that interacts with its occupants and provides them critical information. It incorporates basic camera-based tracking, gesture detection, and reacts chiefly by answering verbal commands. The review of the innumerous other projects involving user-driven interactive spaces is beyond the scope of this paper.

2.3 Story-Driven Interactive Spaces

Narrative and story structures are very common in computer games. However, the number of physically interactive spaces with such elements is very restricted, although the interest in the area has considerably increased in the last years. In this paper, we use the term *story-driven* interactive spaces to refer to spaces that immerse the users in a story that develops partially in response to the users' actions and partially as a consequence of a narrative of events pre-determined by the designers of the space.

Interactive stories and narratives is still a genre to be born, as discussed by Janet Murray in [15]. Murray observes that there are particular narrative genres that seem to be more suitable for interaction such as journey narratives. In particular, Murray studies multi-threaded stories where the user has mechanisms to choose different paths in a web of events, creating an individual, personal story as a result of the interaction.

In the realm of interactive spaces the first experiments were created by Myron Krueger [11], although most of his "narrative" installations portrayed very simple stories. Larry Friedman and Glorianna Davemport's *Wheel of Life* was one of the first works to include complex and rich narrative elements in an interactive space [8]. Naoko Tosa and Ryohei Nakatsu have created two interactive pieces with strong narrative structure [16, 24]. In the *Interactive Poem*, the user dialogues with a stylized female face projected on a large computer screen [24]. In *Romeo and Juliet in Hades*, two users take the roles of Romeo and Juliet after their death and, through speech and gesture interaction, follow their journey towards rediscovering who they were and their love [16].

The work of Aaron Bobick at the MIT Media Laboratory has pushed the sensing envelope of interactive spaces in the recent years. In "*The KidsRoom*" [4], a children bedroom-like space takes a group of children through an adventure covering four different worlds inhabited by friendly monsters. The interaction is very physical, involving running, dancing, jumping, and rowing, completely based on information gathered by cameras.

Our experiments in computer theater (see [19] for a definition of the term), *"SingSong"* [22] and *"It / I"* [21], are also very compelling examples of the possibilities of story-driven interactive spaces. They also differ from most of the examples mentioned above in the emphasis in responsiveness as the key element of creating immersion in a story, instead of the more commonly used device of choice among different story paths. Although choosing the path of a story considerably empowers the user, it is very difficult to assure that all of the paths lead to equally satisfying experiences. In contrast, we have created powerful stories where the user is coerced, through mechanisms not explicitly visible, to remain inside the main path, while carefully designed interaction and good responsiveness keep the illusion that the story is in fact unfolding in response to the user's actions.

3 Centralized vs. Decentralized Control of Story

Most interactive spaces and virtual reality experiences (for instance, [13]) are based on the concept of *exploration*. That is, the user enters a world populated by characters, objects, and other users, and extracts most of its satisfaction from the encounter with the new. There is no narrative structure unfolding during the experience and therefore no need for story representation or control.

The existence of a story in an interactive system requires the management of multiple characters and events in orchestrated ways. A good image of this distinction is to observe that while exploratory worlds have *creatures* living in them, story-based interaction requires *actors*. Actors know that a story must start, develop, reach a climax, and finish.

An important question is where the story is represented and how it is controlled. For example, in most story-based interactive systems created until now, like Perlin and Goldberg [18], Bates et al. [2], and Blumberg and Galyean [3], the story is carried by (semi-) autonomous computer-actors with partial knowledge of the story contents and development. The story is seen as the natural result of the interaction between the characters, the user, and the story traces in the character's brains.

We believe that centralized story control is fundamental for an interactive system to achieve successful story development. As pointed by Langer [12] and Murray [15], well-constructed stories require coordination and synchronicity of events and coincidences. However, the coordination required to achieve coincidence is, in our view, only possible with centralized story control. Also, as brilliantly pointed by Langer in her study of theater ([12], chapter 17), it is essential for dramatic structure to **forecast the future**:

> *"In actual life we usually recognize a distinct situation only when it has reached, or nearly reached, a crisis; but in the theater we see the whole setup of human relationships and conflicting interests long before any abnormal event has occurred...(...) This creates the peculiar tension between the given present and its yet unrealized consequent, "form in suspense", the essential dramatic illusion." (Suzanne Langer,[12] pg. 311).*

If we agree with Langer, it is necessary for an interactive system with dramatic structure to have, in some form, the ability to look ahead and sketch in the present the conflict of the future. To leave this job for each separate character and to expect that they will all come up with the same story structure is pure nonsense.

4 The Story-Character-Device (SCD) Architecture

Interactive spaces are normally complex structures involving multiple inputs, outputs, and control sources. In the interactive space *"It"* described in this paper we have used a novel software architecture called the *story-character-device* architecture, or simply, the *SCD* architecture. Figure 1 shows the basic elements of the SCD architecture that is composed of five levels. The *world level* corresponds to the actual sensors and output generators of the system. The *device level* corresponds to low-level hardware and software to track and recognize gestures and speech from users, and to control the output devices and applications (including low-level control of the movement of computer graphics characters). The *character level* contains software modules that control the actions of the main characters — including here the users and the virtual crew (e.g. cameraman, editor, and light designer) — by considering both the story goals and constraints and the actual events in the real space. The *story level* is responsible for coordinating characters and environment factors, aiming to produce the interactive structure contained in an *interaction script*. Finally, it is possible to have a *meta-story level* that dynamically changes the story as a result of the on-going interaction; such a module should encompass knowledge about how stories are structured and told [5].

The fundamental distinction with previous architecture models [2, 18] is the existence of a story level separated from the character level. As discussed above, we do not believe that story development is achievable simply by character interaction but rather that central control is necessary. On the other hand, preserving the distinction between story and characters simplifies considerably the management of the story. Notice that in the SCD architecture the characters are semi-autonomous since they receive commands from the story control module. However, they coordinate the accomplishment of their goals (as set by the story module) according to the actions happening in the interactive space.

The best analogy to the role of the *story level* in the SCD architecture is the old theatrical figure of a *whisperer* or *prompter*, the person who used to sit underneath a small canopy in the front of stage, facing the actors, and whispering them the next lines to be said. In this model, the actors are responsible for determining **how** the lines are uttered and for making up the accompanying action; the whisperer's job is to assure that story proceeds according to the script, that is, to determine **which** and **when** the lines are said. In the SCD architecture, the story level module watches the user actions (regarding the user as just another character) and tries to adjust the story to match his actions, according to a script that describes how the interaction is supposed to happen. If the meta-story level is present, the script itself can change — as if the printed text in front of an old-timer prompter suddenly shuffled its words into a new plot.

Although the system of our computer theater play *"It / I"* [21] was designed with the concept of SCD architectures in mind (the idea predates the project), our first

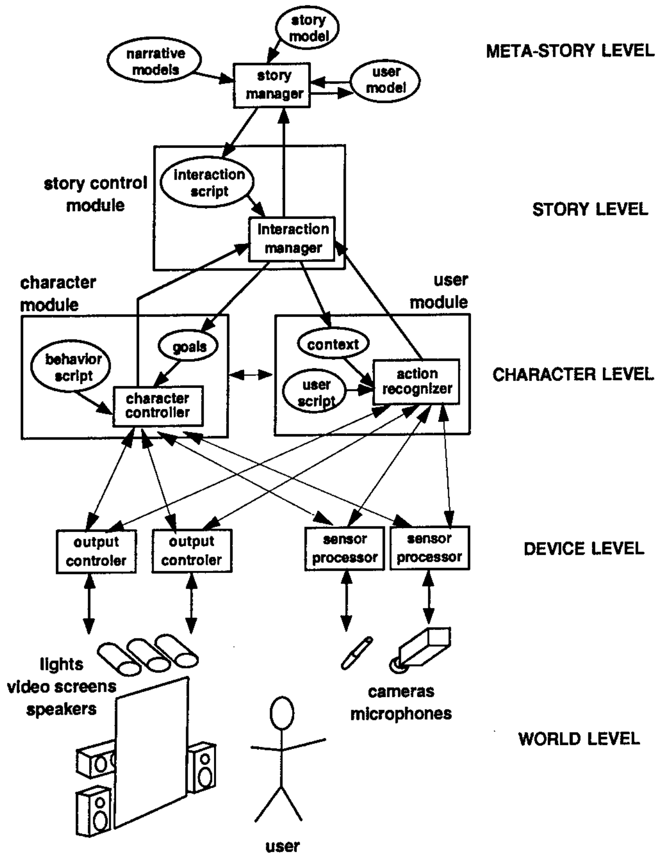


Figure 1. The SCD architecture for interactive story-based systems.

work that fully followed the model is the art installation *"It"* described in the next section. Since the story behind *"It"* is basically the same as the story in *"It / I"*, we can compare the impact of the different architectures in the process of building the systems.

Like most of the work in software architecture, it is hard to characterize the success and even appropriateness of a particular model. We do not claim that SCD is the right or best architecture for a story-based interactive system. However, the comparative experience of building *"It"* and *"It / I"* shows that the SCD model addresses issues that have been neglected by most of previous proposals.

5 *"It"*: An Interactive Space based on the SCD Architecture

In November of 1997 we premiered the computer theater play *"It / I"* at the MIT Media Laboratory, portraying the story of a human character, called ***I***, who was taunted and played by an autonomous computer-graphics character, ***It***. After the

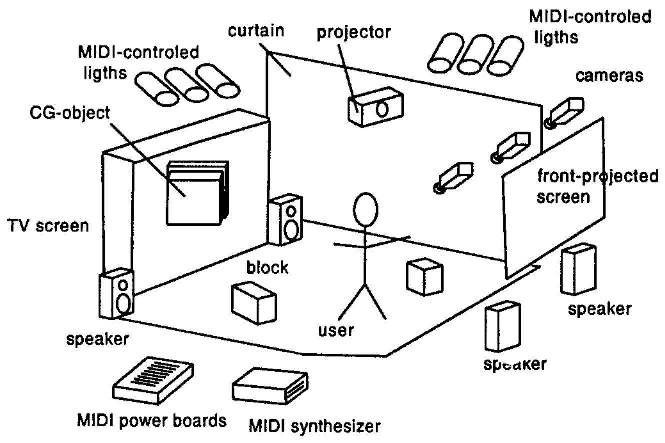


Figure 2. Physical setup of *"It"*.

performances the audience was invited to go up on stage and re-enact the play. This was possible because the stage and the computer character were automatically controlled and therefore the story of the play could be repeated as many times as needed, for each individual member of the audience.

However, we were not totally pleased with the final result of the audience interaction with the computerized story. First we felt that it was necessary to develop further the automatic control system to allow full dramatic immersion of a non-actor. Second, the open stage proved to be a hard place for a member of the audience to fully experience the story of the play, mostly due to the discomfort of being watched.

To overcome these problems we decided to re-create the environment of the play *"It/ I"* in a version for users called *"It"*. Inhabited by the ***It*** character, the space tries to trap the user inside it, under the disguise of a game of taking and showing pictures. *"It"* uses the same visual (images, computer graphics) and sound elements as *"It/I"*, but it was designed to be a self contained, stand-alone piece that can be enjoyed by users completely unfamiliar with the play.

Figure 2 shows the basic physical setup of *"It"*. It consists of two facing screens inside an enclosed, dark room. Three cameras, in a stereo vision configuration, monitor the space detecting the user's presence and position. The installation only admits one user each time. Two independent sets of audio speakers are associated to each screen, reproducing the sound generated by a MIDI synthesizer. Theatrical fixtures controlled by MIDI power devices illuminate the space.

The theme of *"It"* is the same of *"It/I"*, the entrapment of people by technology. However, in *"It"*, we wanted the user to be actually attracted by narrative devices, seduced to play with the machine, gradually start to feel uncomfortable about the situation, and suddenly discover that she has no way to escape. The interactive story is summarized by the following script.

*When there is nobody in the space, **It** (the computer character that inhabits the space) is dormant: green lights, soft machine sound in the background. When **I** (an individual user) enters the space, **It** wakes up, switching the lights on and stopping the music. **It** then brings a camera-like object to the* *right screen. If at any moment* ***I*** *leaves the space,* ***It*** *tries to bring her back by playing loud sound and fast switching color lights. The camera on the right screen follows* ***I*** *around the space and when she stops moving, zooms in and "takes a picture". Immediately after,* ***It*** *moves a TV-like object to the left screen displaying the silhouette of the picture taken by the camera. This silhouette is shown for some moments. The goal of* ***It*** *is to involve* ***I*** *in this game of taking and seeing pictures in a crescendo frenzy of action. Each iteration of the taking-viewing cycle is shorter than the previous. After the game reaches a very fast pace,* ***It*** *looses interest in* ***I****, removes the camera and the TV from the screens, and starts trying to push* ***I*** *out of the space by flickering the lights and simulating on the screen the action of throwing blocks into* ***I****'s direction. When* ***I*** *leaves the space,* ***It*** *goes back to the dormant state, ready for the next victim.*

5.1 The Technology

Most of the technical structure of *"It"* is based on the technology built for *"It / I"* (more details in [20]). To simplify the understanding of the final architecture of *"It"*, we describe briefly the different modules that compose the installation in the following paragraphs.

The vision system module basically tracks the position of the user. In the setup we employed a frontal 3-camera stereo system able to segment the user and to compute the silhouette image that is used to track the position. The stereo system, based on the work of Ivanov et al. [9], constructs off-line a depth map of the background — floor, curtains, and screens. Based on the depth map, it is possible to determine in real-time whether a pixel in the central camera image belongs to the background or to the foreground, in spite of lighting or background screen changes.

The computer graphics modules control the generation and movement of the different objects that appear on the two screens. All the sounds and music in *"It"* are produced by a Korg X5DR MIDI-synthesizer. To control the production of the sounds, the sound module sends the MIDI events corresponding to the different MIDI files associated to each sound and piece of music. To control the lights we employed two 4-channel TOPAZ MIDI-controlled power bricks in a daisy configuration, providing 8 independent dimmer channels. The different channels are controlled by the light module through MIDI signals.

To implement the *"It"* installation we employed a scripting language for interactive spaces called *interval scripts* [20, 22] that is based on the description of the interaction by encapsulating the individual actions in units called *intervals*. Basically, an interval contains descriptions of how to activate and stop the actions of the characters. To describe the narrative flow, the designer can both use procedural methods (like in *Lingo* [1]) but also he can include constraints about what should happen or what can not occur.

5.2 The Architecture of *"It"*

One of the reasons for building *"It"* was to experiment with the idea of implementing an interactive space using the SCD architecture. Figure 3 diagrams the

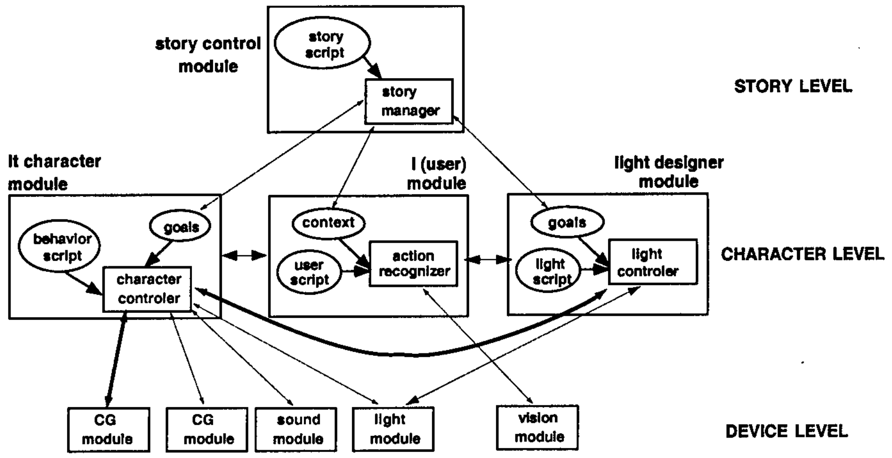


Figure 3. System architecture of *"It"*.

architecture of *"It"*. The device level contains the software modules responsible for the direct control of the computer graphics, sound, lights, and vision system. The character level implements three modules: one corresponding to the ***It*** character, one for the user (or ***I*** character), and a third that corresponds to a "virtual" crewmember, a light designer.

There are two reasons for singling out light control at the character level. First, unlike other actuators such as the computer graphics modules, light is controlled both by the character ***It*** and by the top-level story module. Second, the user module changes its behavior according to the light conditions in the space. For instance, when strong, dark color lighting is used, the positional data becomes unreliable. We found it conceptually easier to have the user module sending a single query just to the light designer module, asking whether the light conditions are normal, instead of checking every dimmer of the low-level dimmer control module.

The basic distinction between the story-level module and the character and crew modules is that the latter contain a great deal of information about how to perform an action while the former is mostly concerned about when and why to do it. For example, the story module has to decide when it is the time for the ***It*** character module to attract ***I***'s attention, when to play ***I*** with pictures, and when to expel him from the space. In this last situation, a request for expelling ***I*** from the space is translated by the ***It*** module into a request to the light designer module to flash the lights from red to white, continuously, and into a sequence of commands to the computer graphics modules to generate CG-explosions of objects on the screens.

5.3 The Experience

We finished the construction of the installation *"It"* in March 1999 in a laboratory space at the MIT Media Laboratory. Since then there have been dozens of users experiencing the feeling of being trapped by ***It***. The current site is far from ideal, since one of the walls is made of glass and there is a considerable amount of

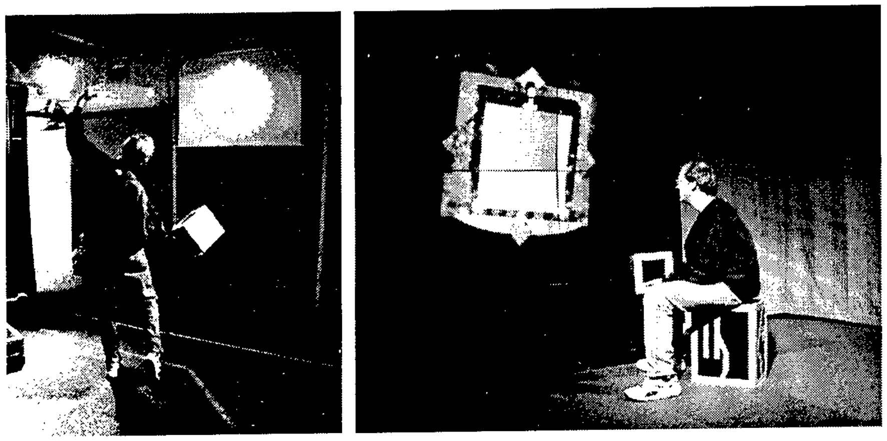


Figure 4. Scenes from a run of *"It"*.

surrounding noise. Nevertheless, we are very pleased with the reaction of the users and with the easiness that the interaction is understood. In particular, there had been some very interesting runs with children. Figure 4 shows pictures of a user playing in *"It"*.

6 Using the SCD Architecture

In order to evaluate the appropriateness of the SCD architecture, we can compare our experience in developing *"It/I"*, where we did not use the paradigm, to the construction of *"It"* which was structured according the SCD model.

The main advantage observed was a reduction in the conceptual complexity of the system. Since the character structure is independent of story control, we could develop each module of *"It"* separately, resulting in more robust and interesting automatic characters. Also, making the ***light designer*** distinct from the other modules allowed the programming of very complex light effects while preserving simple ways to inform the other modules of the lighting conditions. In other words, the SCD architecture allows for easy hiding of complex character or crewmember behaviors.

For instance, there were about 10 different lighting conditions, some of them involving patterned changes of lighting. However, from the standpoint of the ***I*** character, any state of the lights was simply labeled either as "normal" (where lighting conditions allow tracking of detailed features of the body) or "not normal" (when only the presence or absence of the user could be detected). This is, of course, the old concept of data abstraction seen in the context of a story-based system. In many ways, what the SCD model proposes is a good way to structure the abstraction, that is, along the lines between characters and crewmembers and with a supervising story control module.

One of the advantages of a separated story control module felt during the development of *"It"* was the easiness in making changes in the story line. In *"It/I"* such kind of changes many times led to low-level conflicts inside the characters' behavior. In *"It"*, changing the story had no side effects on the characters, allowing much more time and freedom of experimentation with the story. For instance, it becomes trivial to experiment with different ordering for the story events since the character level would respond to the change without the need for alterations.

From an artistic/directorial point of view, the SCD model has the positive benefit of forcing the designer of thinking the interaction as actions being performed by characters. As pointed by, among others, Clurman [6], dramatic structure is best approached if scenes are seen as the result of the conflicting actions and goals of their characters and not as a sequence of emotional states. By isolating the actors from the story and among themselves, the designer has to structure the story as a sequence of commands to each character to act upon the other characters. Although subtle, this distinction benefited tremendously the implementation of *"It"* and helped to design an experience with a stronger dramatic structure.

The major drawback of using the SCD architecture according to our experience with *"It"* is the increase in the number and frequency of messages between modules. Also, it requires a communication protocol where the concepts of "action", "intention", and "goal" can be expressed. In our case this did not become a problem because we employed a very expressive language, ACTSCRIPT, as our communication protocol. ACTSCRIPT was developed by Pinhanez and Bobick (see [20]) based on Roger Schank's *conceptualizations* [23].

7 Conclusion

The main argument of this paper is that centralized control is fundamental in story-driven interactive environments. We do not believe that good story or narrative development can be achieved by dispersing the story through the different characters and actors as proposed in previous works [2, 3, 18]. The SCD architecture is, in this view, just an implementation of the concept that story is the commander of the high-level behavior of the characters.

Our experience using the SCD model in the design and construction of *"It"* has been extremely positive. Although it increases the amount of communication between modules in a system, it reduces the conceptual complexity of designing a story-driven interactive space and increases the flexibility of making changes. We could observe that, compared to *"It / I"*, we did much more experimentation with different developments for the story. Also, as pointed above, the SCD model forces the designer to think of interaction independently of the low-level programming and designing of the characters, facilitating the use of dramatic elements as the key element of the interaction.

Acknowledgements

The ideas presented in this paper benefited immensely from the comments of my Ph.D. advisor, Prof. Aaron Bobick. John Liu implemented the vision system used in *"It"*. The computer graphics objects were created by Raquel Coelho and the sound track was composed by Freedom Baird (originally for *"It / I"*). The production of *"It"* and *"It / I"* was sponsored by the Digital Life Consortium of the MIT Media Laboratory.

References

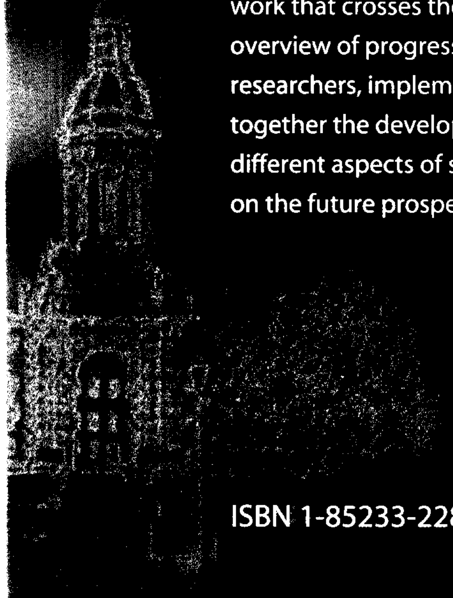
1. Director's User Manual. MacroMind Inc. (1990)
2. Bates, J. A., Loyall, B., Reilly, W. S.: An Architecture for Action, Emotion, and Social Behavior. Proceedings of the Fourth European Workshop on Modeling Autonomous Agents in a Multi-Agent World, S. Martino al Cimino, Italy (1992)
3. Blumberg, B. M., Galyean, T. A.: Multi-Level Direction of Autonomous Agents for Real-Time Virtual Environments. Proc. of SIGGRAPH'95 (1995)
4. Bobick, A. et. alli: The KidsRoom: A Perceptually-Based Interactive and Immersive Story Environment. Presence: Teleoperators and Virtual Environments (1999)
5. Brooks, K. M.: Do Story Agents Use Rocking Chairs? The Theory and Implementation of One Model for Computational Narrative. Proc. of the ACM Multimedia'96 (1996) 1-12
6. Clurman, H.: On Directing. Collier Books, New York, New York (1972)
7. Coen, M. H.: Building Brains for Rooms: Designing Distributed Software Agents. Proc. of IAAI'97, Providence, Connecticut (1997) 971-977
8. Davenport, G., Friedlander, L: Interactive Transformational Environments: Wheel of Life. In: Barrett, E., Redmond, M. (eds): Contextual Media: Multimedia and Interpretation. The MIT Press, Cambridge, Massachusetts (1995) 1-25
9. Ivanov, Y., Bobick, A., Liu, J.: Fast Lighting Independent Background subtraction. Proc. of the IEEE Workshop on Visual Surveillance (VS'98), Bombay, India (1998) 49-55
10. Johnson, M., Wilson, A., Kline, C., Blumberg, B., Bobick, A.: Sympathetic Interfaces: Using a Plush Toy to Direct Synthetic Characters. Proc. of CHI'99, Pittsburgh, Pennsylvania (1999)
11. Krueger, M. W.: Artificial Reality II. Addison-Wesley (1990)
12. Langer, S. K: Feeling and Form. Charles Scribner's Sons, New York, New York (1953)
13. Laurel, B., Strickland, R., Tow, R.: Placeholder: Landscape and Narrative in Virtual Environments. ACM Computer Graphics Quarterly, vol. 28 (2) (1994)
14. Maes, P., Darrell, T., Blumberg, B., Pentland , A.: The ALIVE System: Full-Body Interaction with Autonomous Agents. Proc. of the Computer Animation'95 Conference, Geneva, Switzerland (1995)
15. Murray, J.: Hamlet on the Holodeck: the Future of Narrative in Cyberspace. The Free Press, Simon & Schuster, New York, New York (1997)
16. Nakatsu, R., Tosa, N., Ochi, T.: Interactive Movie System with Multi-person Participation and Anytime Interaction Capabilities. Proc. of ACM Multimedia'98, Bristol, England (1998) 129-137
17. Pentland, A.: Smart Rooms. Scientific American, vol. 274 (4) (1996) 68-76
18. Perlin, K., Goldberg, A.: Improv: A System for Scripting Interactive Actors in Virtual Worlds. Proc. of SIGGRAPH'96 (1996)
19. Pinhanez, C. S.: Computer Theater. Proc. of the Eighth International Symposium on Electronic Arts (ISEA'97), Chicago, Illinois (1997)
20. Pinhanez, C. S.: Representation and Recognition of Action in Interactive Spaces. Ph.D. Thesis. Media Arts and Sciences Program. Massachusetts Institute of Technology (1999)
21. Pinhanez, C. S., Bobick, A. F.: "It / I": A Theater Play Featuring an Autonomous Computer Graphics Character. Proc. of the ACM Multimedia'98 Workshop on Technologies for Interactive Movies, Bristol, England (1999) 22-29
22. Pinhanez, C. S., Mase, K., Bobick, A. F.: Interval Scripts: A Design Paradigm for Story-Based Interactive Systems. Proc. of CHI'97, Atlanta, Georgia (1997) 287-294
23. Schank, R. C., Goldman, N. M., Rieger III, C. J., Riesbeck, C. K: Conceptual Information Processing. North-Holland (1975)
24. Tosa, N., Nakatsu, R.: Interactive Poem System. Proc. of ACM Multimedia'98, Bristol, England (1998) 115-118
25. Weiser, M.: The Computer for the Twenty-First Century. Scientific American (1991) 94-100

Author Index

Managing Interactions in Smart Environments

Research into Smart Buildings and Spaces has increased rapidly over the last few years. Active buildings promise to give greater access and usability to both able bodied and disabled people at work or at home. Smart Environments will perform planning tasks such as suggesting routes through the building to visitors, perception tasks such as gesture and face recognition, action tasks such as calling lifts for robots, and software tasks such as managing the massive event reporting such real-world systems will generate. Facilitating this interaction requires the development of a variety of software infrastructures from support for actuators and sensors, through novel control and interfaces to distributed systems support. Such infrastructures [illegible] isolation of the problem domains. They require research that addresses the interactions between many heterogenous systems: between systems and networks; and between people and systems.

This volume addresses the convergence of research in Distributed Systems, Robotics and Human Centred computing within the domain of smart buildings and present a unique opportunity to investigate work that crosses the boundaries of these disciplines. It provides an overview of progress in a fast-moving area, by bringing together researchers, implementors and practitioners and the papers draw together the developments and concerns of those working on the different aspects of smart environments, as well as providing views on the future prospects for work in this area.



ISBN 1-85233-228-X

ISBN 1-85233-228-X

9 781852 332280

http://www.springer.co.uk